P.S.I.

BASIC AMSTRAD CPC 464, 664 ET 6128

1. MÉTHODES PRATIQUES



J. BOISGONTIER
B. CÉSARD

BASIC AMSTRAD CPC 464, 664 ET 6128

1. MÉTHODES PRATIQUES

Connaissez-vous toute la collection Amstrad chez P.S.I. ?

Pour les Amstrad CPC 464, 664 et 6128 :

Initiation

- La découverte de l'Amstrad — Daniel-Jean David
- Exercices en Basic pour Amstrad — Maurice Charbit

Programmation BASIC

- 102 programmes pour Amstrad — Jacques Deconchat
- Amstrad en famille — Jean-François Sehan
- Super jeux Amstrad — Jean-François Sehan
- Super générateur de caractères sur Amstrad — Jean-François Sehan
- Photographie sur Amstrad et Apple II — Pierrick Moigneau et Xavier de la Tullaye

Maîtrise du BASIC

- Basic Amstrad, 2 — Programmes — Jacques Boisgontier
- Basic Plus, 80 routines sur Amstrad — Michel Martin
- Périphériques et fichiers sur Amstrad — Daniel-Jean David
- Amstrad en musique — Daniel Lemahieu

Assembleur et Pascal

- Assembleur de l'Amstrad — Marcel Henrot
- Turbo Pascal sur Amstrad — Pierre Brandeis et Frédéric Blanc

Système

- Clefs pour Amstrad, 1. Système de base — Daniel Martin
- Le livre de CP/M Plus sur Amstrad 6128 et 8256 — Yvon Dargery
- Clefs pour Amstrad, 2. Système disque — Daniel Martin et Philippe Jadoul
- Clefs pour Amstrad PCW — Eric Baumarti

Intelligence artificielle et graphisme :

- Intelligence artificielle : langage et formes sur Amstrad — Thierry Lévy-Abegnoli et Olivier Magnan
- Animation graphique sur Amstrad — Gilles Fouchard et Jean-Yves Corre
- **Simulation et intelligence artificielle sur Amstrad — René Descamps**

Pour connaître les dernières nouveautés P.S.I.,
ou nous soumettre un problème technique,
nous mettons à votre disposition un service Minitel

Service Minitel
Sur le 3615 tapez OI puis PSI

GROUPE DE LA CITÉ

6-10, boulevard Jourdan, 75014 PARIS
1988

ISBN : 2-86595-230-4

J. BOISGONTIER
B. CESARD

BASIC AMSTRAD CPC 464, 664 ET 6128

1. MÉTHODES PRATIQUES

ÉDITIONS P.S.I.
1988

DU MÊME AUTEUR AUX ÉDITIONS DU PSI :

Le basic de A à Z
Le basic et ses fichiers - Tomes 1 et 2
Basic pour tous
Oric pour tous
52 programmes, Oric pour tous
Le cahier du Basic sur Oric
L'Apple et ses fichiers - Tome 1
Apple pour tous
36 programmes, Apple pour tous
Commodore 64 pour tous (avec Gérard Foucault)
Commodore 64, Méthodes pratiques
MO5 et TO7/70 pour tous
MO5 et TO7/70, Méthodes pratiques
Spectrum pour tous (avec Marcel Henrot)
BASIC MSX, Méthodes pratiques

PRÉSENTATION

Destiné à un public déjà initié à la micro-informatique, le "BASIC AMSTRAD : Méthodes Pratiques" répondra à votre attente de perfectionnement. Jacques Boisgontier, auteur du best-seller "Le Basic et ses fichiers", y adopte une démarche originale en présentant les instructions du Basic AMSTRAD au fur et à mesure de vos besoins.

Si le "BASIC AMSTRAD" ne s'adresse pas aux débutants, il comprend cependant une prise en main suivie des instructions de base, donnant ainsi les notions nécessaires à la compréhension du texte.

Loin d'être une liste sèche d'instructions, cet ouvrage propose quantité de programmes-exemples dans lesquels vous découvrirez de nombreuses astuces utiles.

Cet approfondissement du langage Basic AMSTRAD est complété par un chapitre "programmes" où, du graphisme à la gestion, vous pourrez mettre en application toutes vos connaissances, et créer de très belles pages-écran. A vous de jouer !

SOMMAIRE

PRISE EN MAIN | 1

LE CLAVIER

Si vous possédez un CPC 664, consultez le chapitre 14 page 157. Pour un CPC 6128 consultez le chapitre 15 page 167.

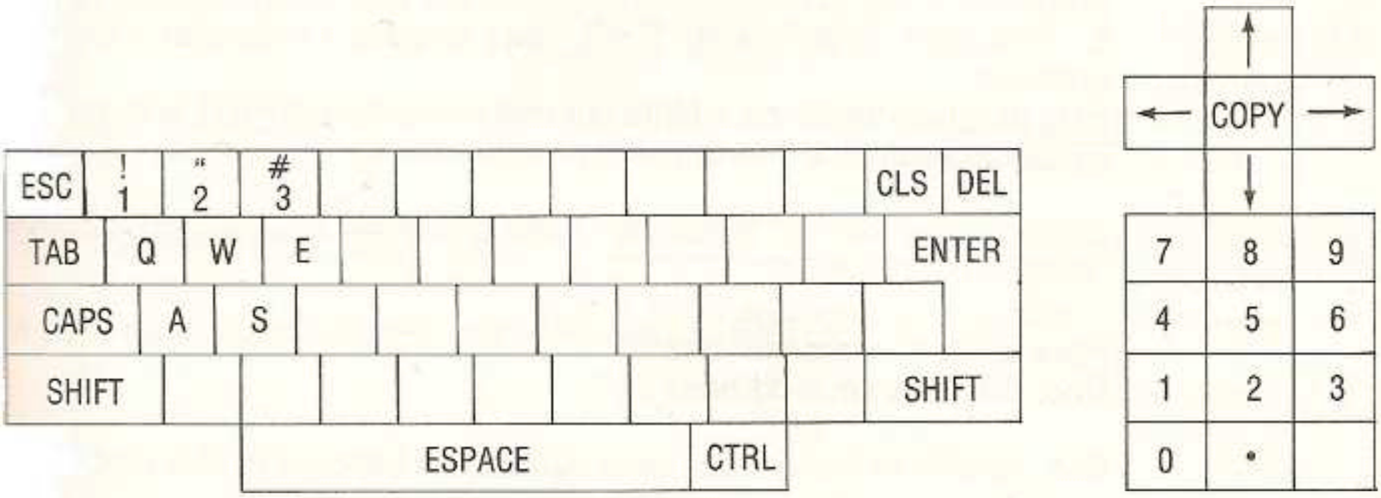
A la mise sous tension, apparaît le message :

```
Amstrad 64K Microcomputer  (v1)

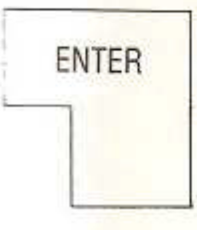
©1984 Amstrad Consumer Electronics Plc

        and Locomotive Software Ltd.

  BASIC 1.0
```



Voyons le rôle des touches essentielles :



Si vous frappez "PRINT 4+5" (afficher 4+5), il ne se passe rien. Pour obtenir un résultat, il faut "valider" la ligne frappée avec la touche ci-contre que nous appelons ENTER.

CAPS

A la mise sous tension, le clavier est en minuscules. Pour le positionner en majuscules, appuyer sur CAPS.
Une seconde pression sur CAPS fait passer en minuscules.

ESPACE

Cette touche, en bas du clavier, déplace le curseur d'une position à droite, sans écrire, permettant ainsi d'introduire des espaces dans le texte frappé.

SHIFT

La touche SHIFT permet d'accéder aux caractères du haut des touches à deux caractères.

- Il faut d'abord appuyer sur cette touche.
- Puis **maintenir** cette touche tout en appuyant sur le caractère désiré.

Exemple : Pour obtenir le caractère "guillemet", appuyer sur les deux touches ci-dessous **simultanément**.

SHIFT — " / 2

Essayez de frapper : PRINT "BONJOUR"

DEL

La touche DEL efface le caractère à gauche du curseur.

CLR

Efface le caractère sous le curseur.

ESC

En appuyant sur cette touche une fois, vous suspendez l'exécution d'un programme qui se poursuit en appuyant sur une touche quelconque.
En appuyant deux fois sur ESC, vous stoppez l'exécution d'un programme.
Si le programme est en attente sur une instruction INPUT, la frappe de ESC une seule fois interrompt le programme.

CTRL

Cette touche permet d'accéder aux caractères dont les codes sont compris entre 1 et 31.

Pour CTRL A, vous obtenez "┌".
Pour CTRL B, vous obtenez "⊥".

Ces caractères ne sont pas ceux obtenus en frappant PRINT CHR$(X).

CTRL SHIFT ESC

Initialise le système (comme une mise sous tension). Il faut appuyer simultanément sur les trois touches.

CTRL ENTER

En appuyant sur CTRL et ENTER (du pavé numérique) le message "RUN" apparaît. Ceci permet de charger et d'exécuter un programme sur cassette.

L'ÉCRITURE DE PROGRAMMES

POUR ÉCRIRE UN PROGRAMME

Il suffit de frapper les instructions en leur affectant un n° de ligne.

Nos des instructions → Touche ENTER

```
10 PRINT "BONJOUR"
20 GOTO 10
run

BONJOUR
BONJOUR
BONJOUR
Break in 20
```

Demande d'exécution → run

Appuyez sur ESC 2 fois pour stopper l'exécution

INSERTION D'UNE LIGNE

Pour insérer une ligne entre 10 et 20, on choisit un n° de ligne intermédiaire (15 par exemple).

```
15 Print "MONSIEUR"

list

10 PRINT "BONJOUR"
15 PRINT "MONSIEUR"
20 GOTO 10
```

SUPPRESSION D'UNE LIGNE

Il suffit de frapper le numéro de ligne suivi de ENTER.

```
15

list
10 PRINT "BONJOUR"
20 GOTO 10
```

La ligne 15 a disparu

- Chaque ligne ne doit pas excéder 255 caractères.
- Tout ce qui suit REM ou ' sur une ligne de programme est du commentaire.
- Plusieurs instructions sur une même ligne doivent être séparées par ":".

L'ÉDITEUR DE PROGRAMME

Il existe deux modes pour modifier une ligne de programme déjà validée par ENTER.

EDIT n° ligne

C'est le mode le plus simple à utiliser. Pour remplacer le caractère "M" par "N" dans la ligne 10, il faut procéder ainsi :

```
10 PRINT "BOMJOUR"
```

- Frapper **EDIT 10**
  La ligne 10 est alors affichée avec le curseur au début de la ligne.

```
10 PRINT "BOMJOUR"
```

- Positionner le curseur **après** le caractère à supprimer avec la flèche → puis appuyer sur la touche **DEL**. Le caractère "M" disparaît.
  Vous pouvez également placer le curseur sur le caractère à supprimer et utiliser la touche **CLR** pour supprimer le caractère.
- Frapper le caractère à insérer (N).
- Valider la ligne modifiée avec la touche ENTER.

La flèche gauche ← permet de déplacer le curseur à gauche et ainsi d'effectuer d'autres corrections sur la même ligne.

Pour annuler une modification en cours, appuyer sur **ESC**.

Pour modifier une ligne en cours de frappe, les règles sont les mêmes.

Lorsqu'une erreur de syntaxe est détectée à l'exécution, BASIC affiche la ligne erronée qui peut ainsi être modifiée en mode EDIT.

MODIFICATION PAR RECOPIE

Ce mode est plus délicat à utiliser. Il permet de recopier une ou plusieurs parties d'une ligne avec la touche COPY, de "sauter" les caractères à supprimer en appuyant sur "SHIFT →".

Pour modifier la ligne : 10 PRINT "BOMJOUR"

- Positionner le curseur sur la ligne 10 en appuyant sur SHIFT et " ↑ " simultanément.
  Il apparaît alors deux curseurs : le curseur **principal** et le curseur de **copie**.

```
curseur de copie —— 10 PRINT "BOMJOUR"
■ ◄—— curseur principal
```

□ Recopier le début de la ligne à modifier avec la touche COPY jusqu'à la lettre "O". "10 PRINT "BO" est alors affiché à la suite du curseur **principal**.

```
10 PRINT "BOMJOUR"
10 PRINT "BO ◄—— ligne recopiée
```

□ "Sauter" la lettre à supprimer en appuyant sur SHIFT et → simultanément.

□ Frapper la nouvelle lettre (N) qui est alors ajoutée à la ligne recopiée.

□ Recopier le reste de la ligne avec la touche COPY.

□ Valider avec ENTER.

La ligne **en cours de recopie** peut être modifiée comme dans le mode EDIT.

Une modification en cours peut être annulée par ESC.

Rien ne vous empêche de composer une ligne à partir de plusieurs lignes.

Le mode d'édition par recopie peut également être utilisé pour modifier une commande mal frappée.

Pour sauvegarder un programme sur cassette ou disquette, utiliser la commande SAVE "nom-programme" (cf. chapitre "Commandes" page 18).

Exemple : SAVE "ESSAI".

La commande "LOAD" permet de lire un programme sur cassette.

Exemple : LOAD "ESSAI".

programme

INSTRUCTIONS DE BASE | 2

LES COMMANDES BASIC

- AUTO
- CAT
- CLEAR
- CHAIN
- CHAIN MERGE
- CONT
- DELETE
- LIST
- LOAD
- MERGE
- NEW
- RENUM
- RUN
- SAVE
- SPEED WRITE
- TRON
- TROFF
- WIDTH

Ces commandes sont acceptées après l'affichage de READY. Elles peuvent aussi être écrites comme instructions dans un programme.

AUTO n° ligne départ, incrément

Génère un numéro de ligne à chaque fois que vous appuyez sur ENTER. On sort du mode "AUTO" en appuyant sur "ESC".

```
AUTO              ' numerote de 10 en 10 a partir de 10
AUTO 100          ' numerote a partir de 100
AUTO 100,5        ' numerote de 5 en 5
```

Si un numéro de ligne existe déjà, le caractère "*" est affiché.

CAT

Donne la liste des programmes et fichiers sur cassette ainsi que leur type.

$: Programme BASIC
% : Programme protégé
* : Programme ou fichier ASCII
& : Programme ou fichier binaire

L'affichage de "OK" signifie que les fichiers ou programmes ont bien été lus.

CLEAR

Efface les variables et tableaux.

CHAIN nom programme, n° ligne

Charge le programme spécifié (normal ou ASCII) et provoque son exécution. L'ancien programme est supprimé.

```
CHAIN "PROG"      ' charge PROG et lance son execution
CHAIN "PROG",100 ' lance l'execution en 100
```

CHAIN MERGE nom programme,n° ligne,DELETE ligne1-ligne2

Concatène le programme spécifié (normal ou ASCII) et provoque son exécution au numéro de ligne spécifié. Les variables ne sont pas remises à zéro.

ON ERROR est annulé, le pointeur de DATA est positionné en début de DATA. DEFINT, DEFREAL et DEFSTR sont annulés.

```
CHAIN MERGE "PROG",100
CHAIN MERGE "PROG",100,DELETE 20
```

CONT

Continue l'exécution d'un programme après arrêt de celui-ci par STOP ou ESC (2 fois).

N'est pas accepté si le programme a été modifié (dans ce cas, utiliser GOTO XX).

DELETE n° ligne début - n° ligne fin

Supprime les lignes de programme entre les limites indiquées.

```
DELETE              ' efface toutes les lignes
DELETE 10-30        ' efface les lignes 10 a 30
DELETE -30          ' efface les lignes jusqu'a 30
```

LIST n° ligne départ - n° ligne fin

Liste les lignes de programme entre les limites indiquées.

```
LIST              ' liste tout le programme
LIST 10-30        ' liste de 10 a 30
LIST -30          ' liste jusqua30
LIST #8           ' liste sur l'imprimante
LIST 10-30,#8
```

LOAD nom-fichier, adresse

Charge le programme spécifié (normal ou ASCII). L'ancien programme en mémoire est effacé.

```
LOAD ""          ' charge le premier programme ou fichier binaire
LOAD "PROG"      ' charge le programme PROG.
```

MERGE nom-programme

Concatène le programme spécifié (sauvegardé en ASCII par SAVE "PROG",A) au programme en mémoire centrale. Les lignes de programmes dont les numéros sont identiques aux numéros des lignes déjà en mémoire remplacent celles-ci.

Cette commande est généralement utilisée pour ajouter des sous-programmes à un programme.

Nous avons observé qu'un programme BASIC normal est également concaténé.

NEW

Efface le programme en mémoire centrale.

RENUM nouveau numéro, première ligne, incrément

"nouveau numéro" qui est par défaut égal à 10 spécifie le premier numéro de la ligne.

"première ligne" spécifie la ligne où doit recommencer la renumérotation. C'est la première ligne par défaut.

"incrément" représente l'incrément à utiliser pour la renumérotation. Il est égal à 10 par défaut.

```
RENUM            ' renumerote de 10 en 10
RENUM 100        ' la Premiere ligne devient 100
RENUM 100,20,5   ' a Partir de 20 de 5 en 5
```

RUN

Exécute le programme en mémoire au premier numéro de ligne. Toutes les variables sont remises à zéro. Les fichiers sont clos.

RUN n° ligne

Exécute le programme à partir de la ligne spécifiée.

```
10 RUN 100          lance l'execution en 100
```

RUN nom-programme

Charge un programme et lance l'exécution.

```
RUN "PROG"          ' charge et execute le programme PROG
RUN ""              ' charge et execute le premier programme
```

SAVE nom-programme, type

Sauvegarde un programme sur cassette. LOAD permet de relire le programme. Si le type est égal à "P", le programme est protégé ; il ne peut être listé. Si le type est égal à "A", le programme est sauvegardé en ASCII, c'est-à-dire sous forme compactée. Il peut alors être ajouté à un autre programme avec la commande MERGE. Le programme peut également être lu comme un fichier séquentiel.

```
SAVE "PROG"         ' sauvegarde le programme sous le nom de PROG
SAVE ""             ' sauvegarde sans donner de nom
SAVE "PROG",P       ' sauvegarde sous forme protegee
SAVE "PROG",A       ' sauvegarde en ASCII (cf MERGE et CHAIN MERGE)
```

SAVE nom-fichier,B,adresse départ,longueur, adresse-exécution

Sauvegarde une partie de la mémoire.

```
SAVE "DES",B,&C000,&4000      sauvegarde l'ecran graphique
```

SPEED WRITE vitesse

Détermine la vitesse de sauvegarde sur cassette (0 ou 1). La vitesse de sauvegarde par défaut est 0 (1000 bauds).

SPEED WRITE 1 spécifie une vitesse de sauvegarde de 2000 bauds.

TRON

Permet de visualiser les numéros des instructions exécutées (cf. chapitre "La mise au point des programmes").

TROFF

Annule 'TRON'.

WIDTH largeur édition

Détermine la largeur d'édition à l'imprimante.

```
WIDTH 70
```

Rappelons que PRINT FRE(0) donne l'espace libre en mémoire centrale. PRINT FRE(" ") réorganise la mémoire.

LES VARIABLES

NOMS

Les noms de variables s'écrivent avec plusieurs lettres ou chiffres mais le premier caractère doit être une lettre.

Un nom de variable ne doit pas être un mot-clé du BASIC.

TYPES

Il y a trois types de variables : les variables **entières réelles,** et les chaînes de caractères. Le type est précisé par un caractère à droite de la variable (%,!,$).

Exemple : X% = 5 (variable du type entier).

Par défaut, lorsque le type n'est pas précisé, les variables sont réelles.

Type	Caractère de déclaration	Exemple
Entières (nombres entiers compris entre − 32768 et + 32767)	%	X%=123: JOURS%= 365
Réelles (6 chiffres significatifs)	!	X! = 123.456
Chaînes de caractères de longueur variable (255 caractères au maximum)	$	NOM$=« DUPONT »

Les variables de même nom avec des types différents sont considérées comme des variables distinctes (A et A$ par exemple).

CLEAR et RUN initialisent les variables avec des valeurs nulles (0 pour les variables numériques, chaîne vide pour les variables chaînes).

Les variables entières sont représentées sur 2 octets et les variables réelles sur 5 octets.

Au-delà de 9 chiffres, les valeurs des variables réelles sont affichées sous forme mantisse-exposant.

CONSTANTES HEXADÉCIMALES ET BINAIRES

Des constantes hexadécimales sont spécifiées par "&h" ou "&" devant le nombre. Des constantes binaires sont définies par "&x" devant le nombre.

```
10 PRINT &A1,&X1101

RUN
 161          13
```

Des constantes hexadécimales définies dans des DATA pourront être lues ainsi :

```
10 DATA &HFF
20 READ x
30 PRINT x

255
```

DEFINT - DEFREAL - DEFSTR

(définition globale de type de variables)

DEFtype lettres Permet de définir globalement le type de toutes les variables dont les noms commencent par les lettres spécifiées, plutôt que de déclarer explicitement le type de chaque variable par un caractère (%, $, !).

On peut cependant déclarer explicitement par un caractère (!, %, $) un type de variable qui aurait été défini par DEFtype. La déclaration explicite est prioritaire.

DEFREAL J Toutes les variables commençant par J et non déclarées explicitement par un caractère (!, %, $) sont des variables réelles.

DEFSTR A-C Toutes les variables commençant par une lettre A, B ou C sont du type chaîne.

DEFINT I-N, R-T Spécifie deux domaines de variables entières.

```
10 DEFINT a
20 a=2.1
30 aa!=2.3              ' declaration explicite prioritaire
40 PRINT a,aa!

2           2.3
```

CONVERSION DE TYPES DE VARIABLES

Le stockage d'une valeur se fait suivant le type de la variable.

```
10 a%=123.45
20 PRINT a%

123
```

LES EXPRESSIONS ET OPÉRATEURS

Une expression peut être simplement une constante du type numérique ou chaîne, une variable ou une combinaison de constantes et de variables liées par des opérateurs.

Exemple : (X − 2) + 4/6

Les opérateurs effectuent des opérations sur des valeurs. Ils sont classés en trois catégories :

1/Arithmétiques
2/Relationnels
3/Logiques

OPÉRATEURS ARITHMÉTIQUES

^	Exponentiation
* /	Multiplication Division
+ −	Addition Soustraction
MOD	Modulo
\	Division entière

L'évaluation des expressions se fait avec l'ordre des priorités des opérateurs défini ci-dessus.

5 + 10/5 est égal à 7

Des parenthèses permettent de changer cet ordre. Ce sont d'abord les expressions entre parenthèses qui sont évaluées.

(5 + 10)/5 est égal à 3

X MOD Y : L'opérateur MOD donne le reste de la division de X par Y.

PRINT 100 MOD 3 → 1

X\Y : Donne la partie entière d'une division. X et Y doivent être compris entre − 32768 et + 32767.

100\3 → 33

OPÉRATEURS RELATIONNELS

Ils comparent 2 valeurs. Le résultat est soit égal à − 1 (condition vraie), soit égal à 0 (condition fausse). Ceci permet de prendre une décision (par IF... THEN... ELSE).

100 IF X<0 THEN PRINT « NOMBRE NEGATIF » ELSE PRINT « NOMBRE POSITIF »

=	Egalité
<	Inférieur
>	Supérieur
<>	Différent
<=	Inférieur ou Egal

OPÉRATEURS LOGIQUES : AND-OR-XOR

Ils testent des relations multiples. Le résultat de l'opération logique est soit faux (égal à 0) soit vrai (égal à − 1) et permet ainsi de prendre une décision par IF... THEN... ELSE.

```
100 IF A<0 AND B<0 THEN PRINT « A et B SONT NEGATIFS »
```

R1	R2	R1 AND R2
V	V	V
V	F	F
F	V	F
F	F	F

R1	R2	R1 OR R2
V	V	V
V	F	V
F	V	V
F	F	F

R1	R2	R1 XOR R2
V	V	F
V	F	V
F	V	V
F	F	F

Les priorités d'évaluation des expressions comportant des opérateurs arithmétiques, relationnels et logiques sont :

1 Parenthèses
2/Opérateurs arithmétiques
3/Opérateurs relationnels (1 seul niveau)
4 Opérateurs logiques

L'exemple ci-dessous utilise directement le résultat d'une comparaison (sans IF) qui est égal à −1 (vrai) ou à 0 (faux).

```
5 '----------- maximum de 2 nombres
10 a=5:b=3
20 mxm=-((a>b)*a+(a<b)*b)
30 PRINT mxm

 5
```

Remarque : La fonction MAX(X,Y) donne directement le maximum de deux nombres.

OPÉRATEURS BOOLEENS

La manipulation de bits et les opérations booléennes sur ces bits s'effectuent avec les opérateurs AND, OR, XOR.

Ces derniers opèrent sur des groupes de 16 bits au plus qui sont spécifiés par des nombres allant de −32768 à +32767 (représentés en complément vrai de façon interne).

Les opérations s'effectuent BIT à BIT.

Exemple :	15	———————>0000000000001111	
	4	———————>0000000000000100	
	15 AND 4	———————>0000000000000100	————>4
Exemple :	4	———————>0000000000000100	
	2	———————>0000000000000010	
	4 OR 2	———————>0000000000000110	————>6
Exemple :	−1	———————>1111111111111111	
	8	———————>0000000000000000	
	−1 AND 8	———————>0000000000001000	————>8

L'ÉCRAN

- MODE
- PEN
- PAPER
- INK
- CLS
- BORDER
- SPEED INK
- LOCATE
- WINDOW
- POS
- VPOS
- WINDOW SWAP

La gestion des couleurs sur AMSTRAD est articulée autour d'une table de correspondance.

Nous vous conseillons de lire attentivement ce chapitre.

C'est volontairement que nous n'avons pas présenté la syntaxe complète de PEN, PAPER et INK en début de chapitre.

MODE numéro

MODE détermine le nombre de caractères par ligne et le type de résolution (cf. chapitre "Graphique haute résolution").

MODE 0 : 20 caractères par ligne
MODE 1 : 40 caractères par ligne
MODE 2 : 80 caractères par ligne

A la mise sous tension, l'écran est en MODE 1.

Frappez en mode direct :

```
MODE 2
```

l'affichage se fait alors en 80 colonnes.

PEN case
PAPER case

Nous expliquons ces instructions pour le MODE 1 :

		couleur
PAPER 0 →	0	1
PEN 1 →	1	24
	2	20
	3	6

Table de correspondance pour MODE 1.

La couleur du stylo (PEN) et du papier (PAPER) se définit à l'aide d'une table de correspondance.

A la mise sous tension, PAPER est égal à 0 et "pointe" vers la case 0 de la table de correspondance (couleur 1) ; PEN est égal à 1 et "pointe" vers la case 1 de la table (couleur 24).

PEN 3 fait pointer vers la case 3 (couleur 6).
PAPER 2 fait pointer vers la case 2 (couleur 20).

Essayez en mode direct :

```
PEN 3
PAPER 2
```

INK case, couleur

Change la couleur d'une case de la table de correspondance. INK 0,26 place la couleur 26 dans la case 0. Si PAPER est égal à 0, la couleur de fond devient blanche.

0	~~1~~	⇐	26
1	24		
2	20		
3	6		

INK 1,0 place la couleur 0 dans la case 1. Si PEN est égal à 1, tout ce qui était écrit en couleur 24 devient noir.

0			
1	~~24~~	⇐	0
2			
3			

Pour le MODE 2, la table de correspondance ne contient plus que 2 cases. Pour le MODE 0, la table comporte 16 cases.

Essayez le programme suivant :

```
10 MODE 0
20 FOR P=1 TO 15
30  PEN P
40  PRINT "stylo";P
50 NEXT P
```

```
stylo 1
stylo 2
stylo 3
stylo 4
```

Il écrit avec les 16 couleurs de la table.

Grâce à INK, la couleur d'un texte déjà écrit peut être modifiée très rapidement. Le programme ci-dessous permute 15 couleurs de 15 lignes de texte.

```
10 '------- ecriture texte
20 MODE 0
30 INK 0,24
40 DIM c(15)
50 FOR p=1 TO 15
60  c(p)=p
70  PEN p
80  PRINT LEFT$("AMSTRAD........",p)
90 NEXT p
100 '---- Permutation couleurs
110 x=c(1)
120 FOR p=1 TO 14
130  c(p)=c(p+1)
140  INK p,c(p)
150 NEXT p
160 c(15)=x:INK 3,c(15)
170 FOR tp=1 TO 400:NEXT tp
180 GOTO 110
```

Le programme ci-dessous affiche 16 traits verticaux représentant les 16 couleurs de la table de correspondance en MODE 0.

```
10 MODE 0
20 FOR p=1 TO 15
30  PLOT p*30,100,p
40  DRAW p*30,200,p
50 NEXT p
```

CLS

Cette instruction permet d'effacer l'écran.

TABLE DES COULEURS

n°	Couleur de l'encre	n°	Couleur de l'encre	n°	Couleur de l'encre
0	Noir	10	Turquoise	19	Vert Marin
1	Bleu	11	Bleu Ciel	20	Turquoise Vif
2	Bleu Vif	12	Jaune	21	Vert Citron
3	Rouge	13	Blanc	22	Vert Pastel
4	Magenta	14	Bleu Pastel	23	Turquoise Pastel
5	Mauve	15	Orange	24	Jaune Vif
6	Rouge Vif	16	Rose	25	Jaune Pastel
7	Pourpre	17	Magenta Pastel	26	Blanc Brillant
8	Magenta Vif	18	Vert Vif		
9	Vert				

TABLE DE CORRESPONDANCE DES COULEURS A LA MISE SOUS TENSION

n° stylo/papier	Mode 0	Couleur Mode 1	Mode 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	1,24	20	1
15	16,11	6	24

Remarque importante : Lorsque vous utilisez PAPER 0 et PEN 1 par exemple, si les deux cases 0 et 1 contiennent la même couleur, le texte affiché n'apparaît pas. Pour le faire apparaître, frapper en mode direct :

```
INK 0,1:  INK 1,24 : PAPER 0 : PEN 1
```

INK case, couleur1, couleur2

Lorsque deux couleurs sont définies dans INK, les deux couleurs sont affichées alternativement.

BORDER couleur1, couleur2

Définit la couleur du pourtour de l'écran. Lorsque deux couleurs sont spécifiées, les couleurs sont affichées alternativement.

SPEED INK case, temps1, temps2

Définit les temps d'affichage pour "couleur1" et "couleur2" lorsque deux couleurs ont été définies dans "INK case, couleur1, couleur2". Le temps est défini en 1/50ᵉ de seconde.

```
10 INK 1,24,6          ' couleur 24 et 6
20 SPEED INK 100,20 ' 1sec et .2sec
```

JEU DE CARACTÈRES

Le programme ci-dessous affiche le jeu de caractères de l'AMSTRAD.

```
10 '------------ jeu de caracteres AMSTRAD
20 FOR c=32 TO 255
30  PRINT CHR$(c);
40 NEXT c
```

Il existe une série de caractères accessibles au clavier avec CTRL.

Le programme suivant affiche la table de codes. On trouvera en annexe la liste des codes ASCII.

```
10 '------------ affichage codes ecran
20 MODE 2
30 debut=31             ' code debut
40 ncol=10              ' nb de colonnes
50 nl=24
60 '
70 FOR l=1 TO nl
80  FOR cl=1 TO ncol
90   cd=(l-1)*ncol+cl+debut:IF cd>255 THEN END
100   PRINT cd;CHR$(cd);
110  NEXT cl
120  PRINT
130 NEXT l
```

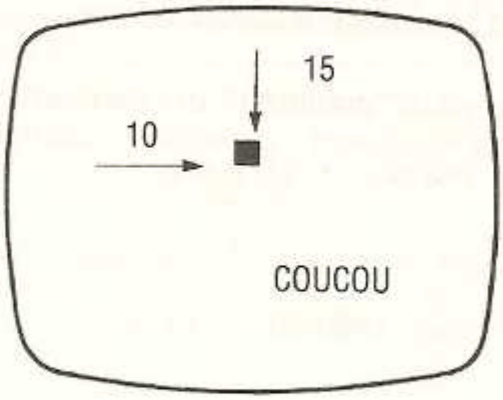
LOCATE X,Y

Positionne le curseur texte dans la colonne X et à la ligne Y.

Pour LOCATE, l'écran est divisé en 25 lignes de 20, 40 ou 80 colonnes suivant le mode défini par l'instruction MODE.

Pour LOCATE, l'origine de l'écran est en haut et à gauche.

```
10 MODE 1
20 LOCATE 10,15              ' colonne 10,ligne 15
30 PRINT "COUCOU"
```

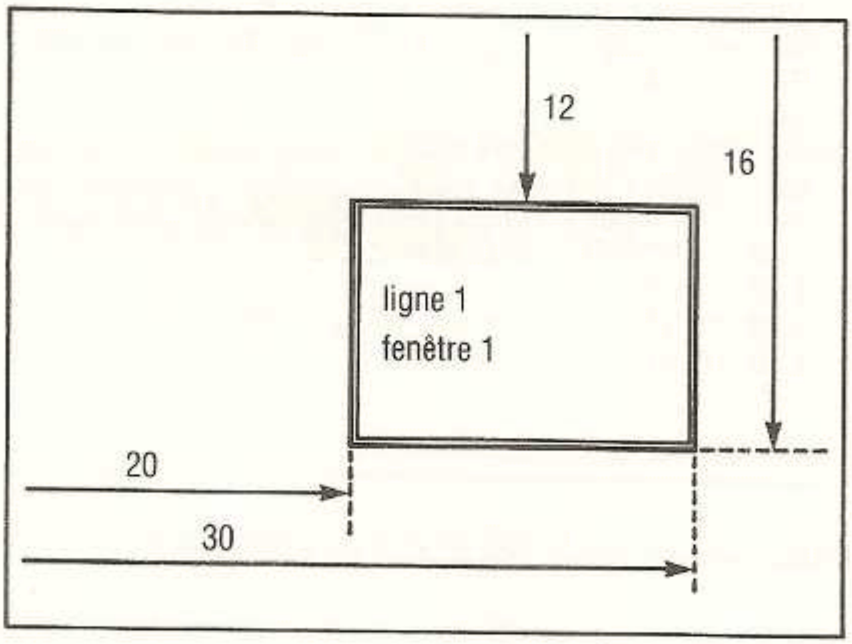


Le programme ci-dessous fait défiler le message « AMSTRAD ».

```
10 '--------- enseigne
20 MODE 1
30 a$="AMSTRAD....."
40 '
50 LOCATE 10,10:PRINT a$
60 a$=RIGHT$(a$,1)+LEFT$(a$,LEN(a$)-1)
70 FOR tp=1 TO 100:NEXT tp
80 GOTO 50
```

WINDOW #numéro fenêtre, gauche, droite, haut, bas

WINDOW définit une fenêtre d'écran. L'écriture se fait seulement dans la fenêtre adressée.




```
10 '------ exemple de fenetre
20 MODE 1
30 WINDOW #1,20,30,12,16
40 PEN #1,1:PRINT #1,"ligne1 fenetre1"
50 PEN #1,3:PRINT #1,"ligne1 fenetre1"
60 '
70 PRINT "ligne1"
80 PRINT "ligne2"
```

Le message "ligne1 fenêtre1" est affiché sur deux lignes.

Les couleurs de stylo, de papier, peuvent être définies pour chaque fenêtre.

Les syntaxes complètes de PEN, PAPER, CLS et LOCATE sont :

PEN # n° fenêtre, stylo
PAPER # n° fenêtre, papier
CLS # n° fenêtre
LOCATE # n° fenêtre,X,Y

POS (# n° fenêtre)
VPOS (# n° fenêtre)

Fournit la position du curseur texte dans la fenêtre spécifiée.

```
10 PRINT "ASDF";
20 PRINT POS(#0)
```

```
10 CLS
20 LOCATE 10,20:PRINT "COUCOU";
30 PRINT VPOS(#0)
```

WINDOW SWAP fenêtre1, fenêtre2

Commute deux fenêtres. Ce qui s'écrivait dans la fenêtre 1 s'écrit dans la fenêtre 2.

```
10 MODE 1
20 WINDOW #1,10,20,10,20
30 WINDOW #2,22,32,2,8
40 PRINT #2,"fenetre2"
50 PRINT #1,"fenetre 1"
60 WINDOW SWAP 1,2
70 PRINT #1,"suite fenetre 1"
```

L'ENTRÉE AU CLAVIER

- INPUT
- LINE INPUT
- INKEY$
- INKEY
- SPEED KEY
- ON BREAK
- KEY
- KEY DEF
- JOY

INPUT "message" ; variable 1, variable 2,...

Permet d'entrer, pendant l'exécution d'un programme, une ou plusieurs valeurs numériques ou chaînes de caractères. Les variables spécifiées dans INPUT sont séparées par des virgules.

Après l'affichage du message, l'opérateur doit entrer les valeurs des variables dans l'ordre défini par l'instruction INPUT, en les séparant par des virgules.

Exemple avec une variable :

```
10 INPUT "Votre nom ";nom$
20 PRINT nom$

run
Votre nom ? DUPONT
DUPONT
```

Exemple avec deux variables :

```
10 INPUT "Nom,age ";nom$,age
15 '
20 PRINT nom$,age

run
Nom,age ? DUPONT,30
DUPONT          30
```

Pour une variable chaîne, si l'opérateur appuie sur ENTER sans entrer de valeur, la variable chaîne devient vide. Une variable numérique prend une valeur égale à 0.

Si l'opérateur entre une chaîne alors que c'est une valeur numérique qui est attendue, le message "**REDO FROM START**" est envoyé.

```
10 INPUT "Age ";age

run
Age ? DUPONT
?Redo from start
Age ?
```

Les chaînes comportant une virgule (séparateur) doivent être entrées **entre guillemets**.

Exemple :

```
Rue? "11,rue NOBEL"
```

INPUT # n° fenêtre, "message" ; variable 1, variable 2,...

Fontionne comme INPUT.

LINE INPUT « MESSAGE » ; variable chaîne

Permet de lire toute une chaîne au clavier, sans tenir compte des séparateurs tels que la virgule comme c'est le cas avec l'instruction INPUT. C'est seulement un retour-chariot qui délimite la fin de la chaîne qui ne peut cependant pas excéder 255 caractères.

```
10 LINE INPUT "Rue? ";rue$
20 PRINT rue$

Rue? 11,rue NOBEL
11,rue NOBEL
```

Remarque : BASIC n'envoie pas de point d'interrogation à la suite du message comme c'est le cas avec 'INPUT'.

INKEY$

INKEY$ lit le clavier en permanence. Si aucun caractère n'a été frappé, la chaîne lue (C$ sur l'exemple) est vide. Le caractère frappé au clavier n'est affiché que si le programme l'a prévu (et non pas automatiquement comme c'est le cas avec INPUT).

Tous les caractères frappés doivent être analysés, y compris le caractère ENTER (code ASCII 13) et le caractère "DEL" (code 127), (avec "INPUT" ces caractères sont gérés par "Basic").

```
10 CLS
20 '
30 c$=INKEY$:IF c$="" THEN 30    ' boucle d'attente
40 PRINT ASC(c$),c$              ' affiche le code et le caractere
50 GOTO 30

65      A
66      B
13
```

Avec INKEY$, le curseur n'apparaît pas.

Ci-dessous, tant que vous n'appuyez pas sur une touche, le message "APPUYER SUR UNE TOUCHE" est affiché.

```
10 C$=INKEY$
20 IF c$<>"" THEN END
30 PRINT "Appuyer sur une touche"
40 GOTO 10

Appuyer sur une touche
Appuyer sur une touche
Appuyer sur une touche
Appuyer sur une touche
Appuyer sur une touche
```

BOUCLE D'ATTENTE

□ Attend que l'opérateur appuie sur une touche quelconque.

```
10 c$=INKEY$:IF c$="" THEN 10
20 PRINT "C'est Parti"
```

□ Teste si l'opérateur répond assez vite.

```
10 PRINT "Repondez (O/N) (vite) "
20 h=TIME
30 '
40 r$=INKEY$:IF r$<>"" THEN 80
50 IF TIME>h+1000 THEN PRINT "Trop tard":END
60 GOTO 40
70 '
80 ' suite
```

Le programme ci-dessous saisit une chaîne de caractères. Les caractères frappés peuvent être contrôlés dès leur frappe sans attendre la validation par ENTER.

```
10 '---------- saisie d'une ligne avec INKEY$
30 MODE 2
40 xs=10:ys=10              ' coordonnees saisie
50 GOSUB 80
60 PRINT:PRINT lig$
65 END
70 '------------------------- spgm saisie dans lig$
80 lig$=""
90 '
100 lg=LEN(lig$):LOCATE xs+lg,ys:PRINT CHR$(143);CHR$(32)  ' 1
43:curseur
110 LOCATE xs+lg,ys
120 '
130 c$=INKEY$:IF c$="" THEN 130              ' attente frappe
140 c=ASC(C$)
150 '
160 IF c<>127 THEN 190                       ' code suppression
170 IF lg>0 THEN lig$=LEFT$(lig$,lg-1):PRINT CHR$(8);CHR$(32):
GOTO 100 ELSE 100
180 '
```

→

```
190 IF c=13 THEN 250                          ' code ENTER?
200 IF c<32 OR c>127 THEN PRINT CHR$(7);:GOTO 130
210 lig$=lig$+c$                              ' ajout caractere
220 PRINT c$                                  ' affichage caract.
230 GOTO 100
240 '
250 LOCATE xs+lg,ys:PRINT CHR$(32)
260 RETURN
```

INKEY (n° touche)

Permet de tester si une touche est enfoncée.

Lorsque l'utilisateur appuie sur la touche spécifiée, la valeur obtenue est égale à 0.

Si l'utilisateur appuie simultanément sur une touche et SHIFT ou CTRL ou SHIFT et CTRL, on obtient :

touche non enfoncée	**−1**
touche seule	**0**
touche + SHIFT	**32**
touche + CTRL	**128**
touche + SHIFT + CTRL	**160**

```
10 PRINT "APPUYEZ SUR 'A'"
20 '
30 IF INKEY(69)=0 THEN PRINT "Vous appuyez sur A"
40 IF INKEY(69)=32 THEN PRINT "Vous appuyez sur A et SHIFT"
50 IF INKEY(69)=128 THEN PRINT "Vous appuyez sur CTRL et A"
60 GOTO 30
```

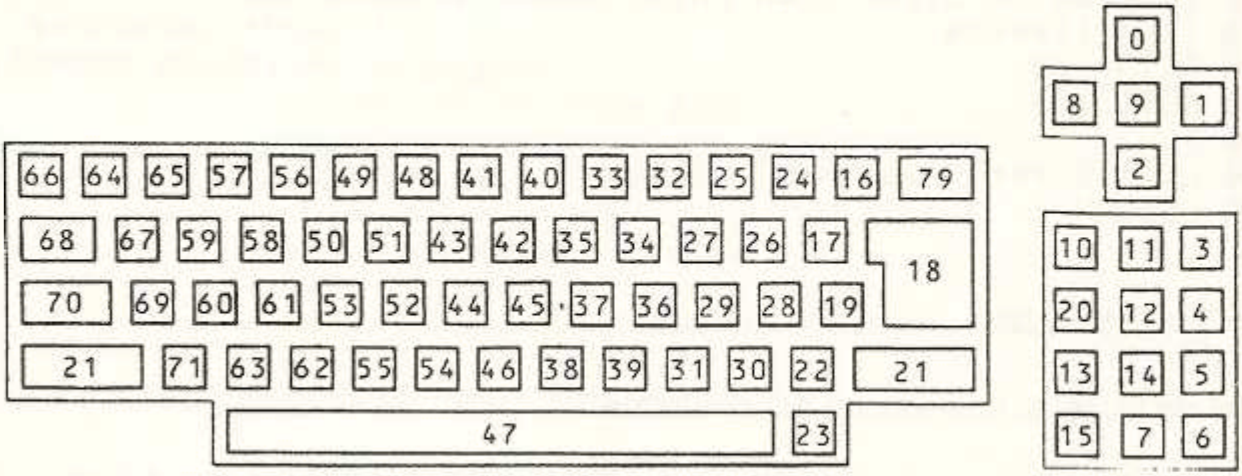
INKEY ne tient pas compte du délai de répétition comme INKEY$. Plusieurs touches enfoncées peuvent être testées simultanément.

Le programme ci-dessous permet de rechercher le numéro d'une touche.

```
10 '--------------- fonction INKEY(touche)
20 PRINT "Appuyez sur une touche"
30 FOR i=0 TO 79
40  c=INKEY(i)
50  IF c=0 THEN PRINT "code=";i       ' touche enfoncee?
60 NEXT i
70 GOTO 30
```

Les numéros de touche sont distincts des codes ASCII des caractères associés. Les caractères associés aux touches peuvent être redéfinis par KEYDEF.

NUMÉROS DES TOUCHES DU CLAVIER :



Le programme "**télécran**" ci-dessous déplace un point à l'écran en laissant une "trace" du passage, permettant ainsi de réaliser des dessins.

```
10 ' telecran
15 '
20 LOCATE 1,22:PRINT "utiliser les fleches
30 MODE 1
40 '
50 x=200:y=200
60 '
70 IF INKEY(8)=0 THEN x=x-1
80 IF INKEY(1)=0 THEN x=x+1
90 IF INKEY(0)=0 THEN y=y+1
100 IF INKEY(2)=0 THEN y=y-1
110 '
120 PLOT x,y,1
130 GOTO 70
```

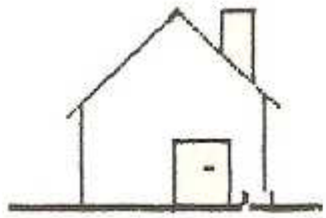
Pour obtenir un déplacement dans les directions diagonales, appuyer sur deux flèches simultanément.

Nous avons écrit le programme de télécran ci-dessus d'une autre façon. L'expression (INKEY(8)=0) s'évalue en −1 si le test est vrai et en 0 si le test est faux. En la combinant à (INKEY(1)=0) nous obtenons le déplacement du curseur sur l'axe horizontal.

```
10 '------------- telecran avec fleches
20 MODE 1
30 LOCATE 1,22:PRINT "Utiliser les fleches"
40 x=200:y=200
50 '
60 x=x+(INKEY(8)=0)-(INKEY(1)=0)
70 y=y-(INKEY(0)=0)+(INKEY(2)=0)
80 '
90 PLOT x,y,1
100 GOTO 60
```

Le programme "télécran" ci-dessous permet de dessiner et d'effacer dans 8 directions. La couleur se choisit avec 1, 2, 3.

```
10 '------------------- telecran 8 directions + couleurs
20 MODE 1:INK 0,24:INK 1,1:PAPER 0:PEN 1
30 ce=1:cf=0                   ' ecriture/fond
40 SPEED KEY 10,2              ' vitesse clavier
50 LOCATE 1,20:PRINT "fleches pour deplacer"
60 LOCATE 1,21:PRINT "L:lever B:baisser E:effacer  Couleurs: 1,2,3"
70 x=200:y=200
80 '------------------------ curseur clignotant
90 t=TEST(x,y)
100 '
110 c$=INKEY$:IF c$<>"" THEN 150
120 PLOT x,y,ce:PLOT x,y,cf
130 GOTO 110
140 '
150 IF l=0 THEN PLOT x,y,ce
160 IF l=1 THEN PLOT x,y,t
170 IF l=2 THEN PLOT x,y,cf
180 '
190 IF INKEY(8)=0 THEN IF x>2 THEN X=X-2
200 IF INKEY(1)=0 THEN IF x<600 THEN X=X+2
210 IF INKEY(0)=0 THEN IF y<397 THEN Y=Y+2
220 IF INKEY(2)=0 THEN IF y>2 THEN Y=Y-2
230 '
240 c$=UPPER$(c$)
250 IF c$="L" THEN l=1          ' lever
260 IF c$="B" THEN l=0          ' baisser
270 IF c$="E" THEN l=2          ' effacer
280 IF VAL(c$)<>0 THEN ce=VAL(c$)
290 GOTO 90
```



SPEED KEY délai, intervalle répétition

Règle le délai de répétition des touches et la vitesse de répétition en 1/50e de seconde.

```
SPEED KEY 10,20
```

A la mise sous tension, SPEED KEY est égal à 10,3.

ON BREAK GOSUB n° ligne

Définit l'adresse d'un sous-programme vers lequel il y aura branchement si l'opérateur appuie sur "ESC" deux fois.

```
10 '------------- ON BREAK GOSUB
20 ON BREAK GOSUB 100
30 PRINT "Appuyer sur ESCAPE 2 fois"
40 GOTO 30
90 '----------- spgm ON BREAK
100 INPUT "On arrete vraiment (O/N) ";r$
105 IF r$="O" THEN END
120 RETURN
```

ON BREAK STOP

Annule l'effet de "ON BREAK GOSUB".

```
10 '------------- ON BREAK STOP
20 ON BREAK GOSUB 100
30 PRINT "Appuyer sur ESCAPE 2 fois"
40 GOTO 30
90 '---------- sp9m ON BREAK
100 INPUT "Annulation ON BREAK (O/N) ";r$
105 IF r$="O" THEN ON BREAK STOP
120 RETURN
```

KEY numéro, CHR$(n) + chaîne + CHR$(n)

Associe une chaîne de caractères à une touche dite de fonction. Lorsque vous appuyez sur la touche de fonction, la chaîne associée à la touche est affichée, facilitant ainsi l'écriture des programmes.

Si la chaîne est suivie de CHR$(13), la commande codée dans la chaîne est exécutée.

Par exemple, si vous écrivez :

```
KEY 128,"list"+CHR$(13)
```

il suffit d'appuyer sur le 0 du "pavé" numérique pour obtenir la liste du programme.

La chaîne définie dans KEY ne doit pas dépasser 32 caractères et le total des chaînes pour l'ensemble des touches de fonction ne doit pas être supérieur à 100.

Les numéros des 12 touches du "pavé" numérique vont de 128 à 140.

0 : 128	7 : 135
1 : 129	8 : 136
2 : 130	9 : 137
3 : 131	. : 138
4 : 132	ENTER : 139
5 : 133	CTRL/ENTER : 140
6 : 134	

Exemples divers :

```
10 '------------ touche de fonction
20 KEY 138,"CLS"                  ' . du clavier numerique
30 '--------
40 KEY 137,"cls"+CHR$(13)         ' 9 du clavier numerique
50 '--
60 KEY 128,CHR$(13)+"cls:ink 1,0:ink 1,24"+CHR$(13)  ' 0 numerique
```

KEF DEF n° touche, répétition, code normal, code SHIFT, code CTRL

Redéfinit les codes ASCII d'une touche.

L'exemple ci-dessous redéfinit la touche M (n° 38). En appuyant sur "M", c'est la lettre "A" qui est affichée. En appuyant sur "M" et "SHIFT", c'est la lettre "B" qui est affichée.

```
10 '---------- redefinition touche M
20 KEY DEF 38,1,65,66,67    ' A(normal),B(shift),C(control) Pour
   touche M
```

"répétition" égal à 1 spécifie que la touche est à répétition.

Pour connaître le numéro de touche, cf. "INKEY".

JOY (0 ou 1)

Deux joysticks peuvent être connectés sur AMSTRAD.

Joystick(n) donne la direction du manche de joystick. Les valeurs décimales obtenues sont :

Direction	Valeur décimale	Bit
HAUT	1	0
BAS	2	1
GAUCHE	4	2
DROITE	8	3
FEU1	16	4
FEU2	32	5

```
10 PRINT JOY(0)
20 GOTO 10

run
1
1
8
8
```

Les joysticks peuvent également être lus comme les touches du clavier par INKEY$ ou INKEY (n° touche). La figure ci-dessous donne les numéros de touche correspondant aux joysticks.

touche n° 72 ↑
n° 74 ← → 75
↓ 73

FEU1: 76
FEU2: 77

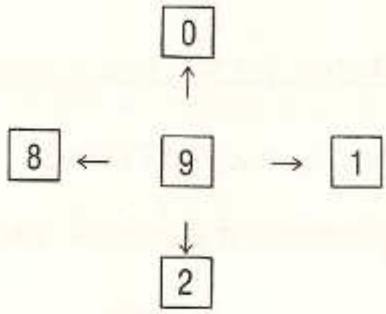
48 ↑
50 ← → 51
↓ 49

FEU1: 52
FEU2: 53

```
5 '------------- telecran avec joystick
10 MODE 1
50 x=200:y=200
60 '
70 IF INKEY(74)=0 THEN x=x-1
80 IF INKEY(75)=0 THEN x=x+1
90 IF INKEY(72)=0 THEN y=y+1
100 IF INKEY(73)=0 THEN y=y-1
110 '
120 PLOT x,y,1
130 GOTO 70
```

Sans joystick, on pourra utiliser les flèches du clavier.

Les numéros de touches sont :



La touche "COPY" peut être utilisée comme touche de mise à feu.

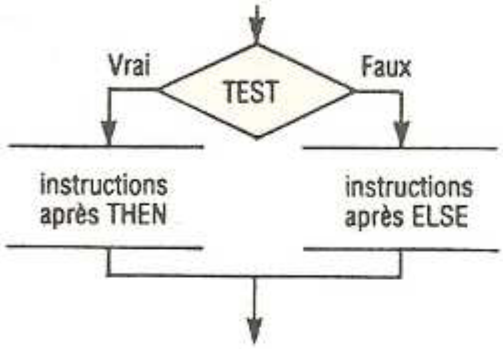
LES TESTS

- **IF...THEN...ELSE**
  SI...ALORS...SINON

IF...THEN...ELSE

IF expression logique — vraie ou fausse
THEN suite d'instructions — si expression logique vraie
ELSE suite d'instructions — si expression logique fausse

Cette instruction teste si une expression logique est vraie ou fausse. Si celle-ci est vraie, alors toutes les instructions comprises entre THEN et ELSE, sont exécutées. Sinon, ce sont toutes les instructions après le ELSE qui sont exécutées.




```
10 INPUT "Nombre A,Nombre B ";a,b
30 IF a>b THEN PRINT "A>B" ELSE PRINT "A<=B"
40 GOTO 10

run
Nombre A,Nombre B ? 3,5
A<=B
Nombre A,Nombre B ? 6,2
A>B
```

En fait, le test peut se faire, non seulement sur une expression logique, mais aussi sur une expression arithmétique qui est interprétée comme fausse si elle a une valeur nulle ou comme vraie pour toute autre valeur. Mais on évitera d'utiliser cette particularité du BASIC.

```
5 i=4
10 IF i THEN PRINT "I est different de 0"
```

```
5 i=120
10 IF (i>10)*(i<20) THEN PRINT "I est compris entre 10 et 20"
```

Les IF...THEN...ELSE... peuvent être **emboîtés**, mais il faut alors bien s'assurer qu'à chaque IF-THEN il correspond un ELSE.

```
10 INPUT "Nombre A,Nombre B";a,b
20 IF a<=b THEN IF a<b THEN PRINT "A<B" ELSE PRINT "A=B" ELSE PRINT
"A>B"
```

LA BOUCLE AUTOMATIQUE

- FOR/NEXT
- WHILE/WEND

FOR variable-compteur = valeur début TO valeur fin STEP pas
instruction 1
instruction 2
NEXT variable compteur

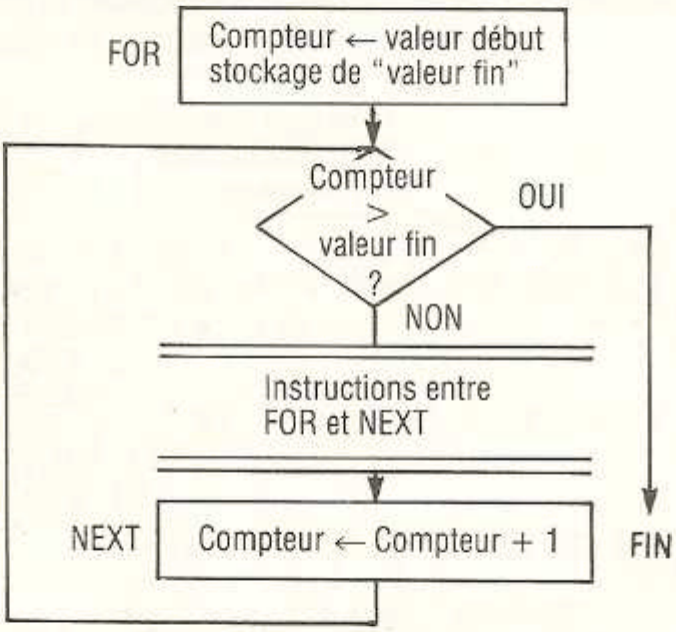
Permet d'écrire des boucles de façon plus concise.

Lorsque l'instruction **FOR** est exécutée, le BASIC affecte à la **variable-compteur** la **valeur début** spécifiée et mémorise la **valeur fin** indiquée après **TO**.

Toutes les instructions entre FOR et NEXT sont d'abord exécutées avec **variable-compteur = valeur début**. L'exécution de NEXT augmente la valeur de la variable-compteur du **pas** spécifié dans **STEP** (1 par défaut).

Si la valeur de variable-compteur est inférieure ou égale (pour un STEP positif) à **valeur fin**, l'exécution se poursuit à l'instruction après FOR. Par conséquent, les instructions entre FOR et NEXT sont à nouveau exécutées avec la **nouvelle valeur** de la **variable-compteur**.

Si la valeur de la variable-compteur dépasse **valeur fin** (pour un pas positif), l'exécution de la boucle s'achève et **le programme se poursuit après NEXT**.



A la fin de la boucle, la variable-compteur a une valeur égale à **valeur fin + 1**.

Exemples :

```
10 FOR c=1 TO 4 'Pour c=1 jusqu'a 4
20  PRINT c,c*c
30 NEXT c       'augmenter c Puis aller aPres for si c<4
40 PRINT "fin";c
```

```
 1              1
 2              4
 3              9
 4              16
fin 5
```

```
10 '----------exemPle avec steP negatif----------
20 FOR c=4 TO 1 STEP -1
30   PRINT c;c*c
40 NEXT c
50 PRINT c;"fin"
```

```
 4              16
 3              9
 2              4
 1              1
 0 fin
```

Calculées au moment de l'exécution de l'instruction FOR, donc une seule fois "**valeur début**", "**valeur fin**" et "**pas**" ne varient pas en cours d'exécution de la boucle si les valeurs des variables qui ont servi à les calculer évoluent pendant l'exécution. En revanche, la valeur de la variable compteur peut être modifiée.

```
10 f=1
20 x=10
30 FOR i=1 TO x
40  IF i=5 AND f=1 THEN x=8
50  PRINT i;
60 NEXT i

 1  2  3  4  5  6  7  8  9  10
```

Ne peut être modifié pendant l'exécution

```
10 FOR i=1 TO 5
20  PRINT i;
30  INPUT "nombre ";a(i)
40  IF a(i)<0 THEN PRINT "erreur" : i=i-1
50 NEXT i
1 nombre ? 4
2 nombre ? -2
erreur
2 nombre ? 3
3 nombre ? 5
```

On revient en arrière

BOUCLES EMBOÎTÉES

Plusieurs boucles FOR... NEXT peuvent être "emboîtées", c'est-à-dire qu'une boucle peut être placée à l'intérieur d'une autre. Mais il est interdit de les faire se chevaucher.

Exemple :

```
10 FOR i=1 TO 3
20  PRINT"i= ";i;"j= ";
30   FOR j=1 TO 8
40    PRINT J;
50   NEXT j
60  PRINT
70 NEXT i


i= 1 j= 1  2  3  4  5  6  7  8
i= 2 j= 1  2  3  4  5  6  7  8
i= 3 j= 1  2  3  4  5  6  7  8
```

S'il n'y a pas d'instruction entre NEXT J et NEXT I, NEXT J,I donne le même résultat. NEXT (au lieu de NEXT J) est accepté, puisqu'en fait NEXT incrémente le compteur du FOR le plus récent et que celui-ci est supprimé dès qu'il atteint la valeur limite. Mais pour des raisons de lisibilité, on indiquera le nom de la variable.

SORTIE D'UNE BOUCLE FOR

On peut sortir d'une boucle FOR par "GOTO" sans problème. (Les versions Microsoft antérieures posaient un problème dans le cas où un indice non "épuisé" était utilisé dans une autre boucle FOR "interne").

```
10 FOR i=1 TO 5
20   IF i=3 THEN GOTO 50    'sortie par goto
30 NEXT i
40 '
50 FOR j=1 TO 3
60  FOR i=1 TO 4            'i utilise a nouveau
70   PRINT "i= ";i;
80  NEXT i
85  PRINT
90 NEXT j
100 PRINT FRE(0)
110 GOTO 10
          i= 1 i= 2 i= 3 i= 4
          i= 1 i= 2 i= 3 i= 4
          i= 1 i= 2 i= 3 i= 4
           43349
          i= 1 i= 2 i= 3 i= 4
          i= 1 i= 2 i= 3 i= 4
          i= 1 i= 2 i= 3 i= 4
           43349
          i= 1 i= 2 i= 3 i= 4
```

WHILE condition
instructions
WEND

Exécute les instructions entre WHILE et WEND tant que "condition" est vraie.

Ci-dessous, le message "Appuyez sur une touche" est envoyé tant que l'opérateur n'appuie pas sur une touche.

```
10 WHILE INKEY$=""     ' tant que INKEY$ est vide
20  PRINT "Appuyez sur une touche"
30 WEND

run
Appuyez sur une touche
Appuyez sur une touche
Appuyez sur une touche
```

TRAITEMENT DES DONNÉES | 3

LES DONNÉES

- **DATA**
- **READ**
- **RESTORE**

DATA

L'instruction DATA permet de définir des données dans le programme lui-même. Celles-ci sont ensuite lues dans des variables par l'instruction "READ variable" (LIRE variable).

```
10 DATA 10,16,13,14
15 '
20 READ N1
30 READ N2
40 READ N3
50 READ N4
60 '
70 PRINT N1,N2,N3,N4
80 PRINT "MOYENNE :";(N1+N2+N3+N4)/4

 10            16            13            14
MOYENNE : 13.25
```

Pointeur de DATA au départ

Pointeur de DATA après 2 READ exécutés

READ

"READ N1" lit la première donnée (10) dans N1.
Le pointeur de DATA (géré par BASIC) progresse de 1. Ainsi "READ N2" lit la deuxième donnée dans N2, etc...

Les données peuvent être écrites sur plusieurs lignes :

```
10 DATA 10,16
20 DATA 13,14
```

Les lignes 10 et 20 sont équivalentes à la ligne :

```
10 DATA 10,16,13,14
```

L'implantation des DATAs dans un programme n'a pas d'importance. Elles sont lues dans l'ordre de la numérotation.

Les chaînes de caractères comportant des caractères spéciaux doivent être placées entre guillemets.

Sur l'exemple ci-dessous, sans la présence de guillemets, la virgule serait considérée comme séparateur.

```
10 DATA "8,rue de Provence"
20 '
30 READ x$
40 PRINT x$

8,rue de Provence
```

RESTORE

Positionne en début de DATA ce qui permet de relire les données depuis le début.

```
10 DATA 6,3,14
20 '
30 READ A,B,C                    'PREMIERE LECTURE
40 '
50 RESTORE                       'DEBUT DATA
60 '
70 READ D,E,F                    'DEUXIEME LECTURE
80 '
90 PRINT A,B,C
100 PRINT D,E,F

   6            3            14
   6            3            14
```

RESTORE n° ligne

Positionne sur le numéro de ligne de DATA spécifié vous permettant ainsi de sélectionner des données particulières.

```
10 DATA 6,3,14
20 DATA 4,2,8
30 RESTORE 20              'LIGNE 20
40 READ X,Y,Z
50 PRINT X,Y,Z

 4            2            8
```

ERREURS

■ DATA EXHAUSTED :

Si le nombre de READ exécutés est supérieur au nombre de données en DATA, le message "DATA EXHAUSTED" apparaît.

```
10 DATA 15,10   <- Il manque une donnée
15 '
20 READ X
30 READ Y
40 READ Z

DATA exhausted in 40
```

■ SYNTAX ERROR :

Le type de la donnée lue doit s'accorder avec le type de la variable.

```
10 DATA NICOLAS   <- Chaîne
15 '
20 READ X   <- Numérique

Syntax error in 10
```

L'exemple ci-dessous lit un nom au hasard dans une liste de quatre noms.

```
10 DATA NICOLAS,SOPHIE,JACQUES,BRUNO
20 '
30 RESTORE
40 X=INT(RND*4)
50 '
60 FOR I=1 TO X:READ X$:NEXT I
70 '
80 READ NOM$
100 PRINT NOM$
110 GOTO 30

  NICOLAS
  BRUNO
  JACQUES
```

Ci-dessous, nous sélectionnons un groupe de DATA. Chaque groupe peut comporter plusieurs lignes. La fin de chaque groupe est repérée par le caractère "*".

```
10 DATA MAISON,PORTE,CHAMBRE
20 DATA *
30 DATA JEAN,PIERRE,PAUL,JACQUES
40 DATA *
50 DATA ROUE,CADRE,FREIN,PEDALE,SELLE
60 DATA *
70 '
80 RESTORE
90 INPUT "QUEL GROUPE (1,2,3)";G
95 '
100 IF G=1 THEN 200
105 '
110 READ X$:IF X$="*" THEN G=G-1: GOTO 100
120 GOTO 110
130 '
200 READ MOT$
210 PRINT MOT$
220 GOTO 80

 QUEL GROUPE (1,2,3) ?  2

JEAN
 QUEL GROUPE (1,2,3) ?  1
MAISON
 QUEL GROUPE (1,2,3) ?  3
ROUE
```

LES TABLES

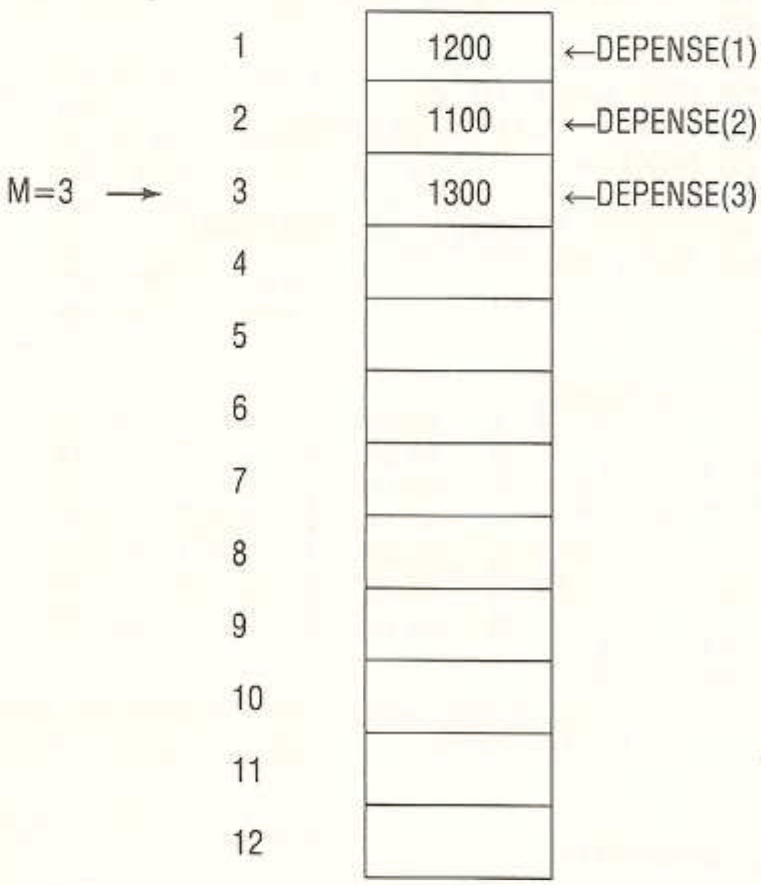
Les tables contiennent des éléments de même nature auxquels nous accédons par un indice. Nous pouvons ainsi traiter les éléments d'une table en faisant simplement varier un indice.

Une table comportant plus de 10 éléments doit être dimensionnée par :

DIM nom-table (nombre éléments)

Soit une table des dépenses relatives aux douze mois de l'année. Les éléments de cette table que nous appelons DEPENSE() sont connus sous les noms de DEPENSE(1), DEPENSE(2),..., DEPENSE(12).

TABLE DEPENSE()



Pour documenter la table, nous faisons varier un indice M de 1 à 12.

```
10 DIM depense(12)
20 '
30 FOR m=1 TO 12
40  PRINT "Mois ";m;
50  INPUT depense(m)
60 NEXT m

run
Mois  1 ? 1200
Mois  2 ? 1100
Mois  3 ? 1300
........
Mois  12 ? 1500
```

Au départ, l'indice M étant égal à 1, l'instruction INPUT DEPENSE(M) est équivalente à INPUT DEPENSE(1). Par conséquent, c'est dans l'élément DEPENSE(1) que la première valeur est introduite. Au second passage dans la boucle, M étant égal à 2, c'est dans DEPENSE(2) que la seconde valeur est introduite. Etc.

Pour connaître le total des dépenses entre deux mois M1 et M2, nous écrivons :

```
10 DIM depense(12)
20 '
30 FOR m=1 TO 12
40  PRINT "Mois ";m;
50  INPUT depense(m)
60 NEXT m
70 '---------------
90 PRINT
100 INPUT "Mois 1";m1
110 INPUT "Mois 2";m2
120 '
140 ttal=0
150 FOR m=m1 TO m2
160  ttal=ttal+depense(m)
170 NEXT m
180 PRINT
190 PRINT "Total:";m1;m2;ttal
200 GOTO 90

run

saisie table

Mois 1? 1
Mois 2? 12

Total: 1  12 14600

Mois 1? 1
Mois 2? 6

Total: 1  6  7200
```

Il existe pour les tables un élément 0 :

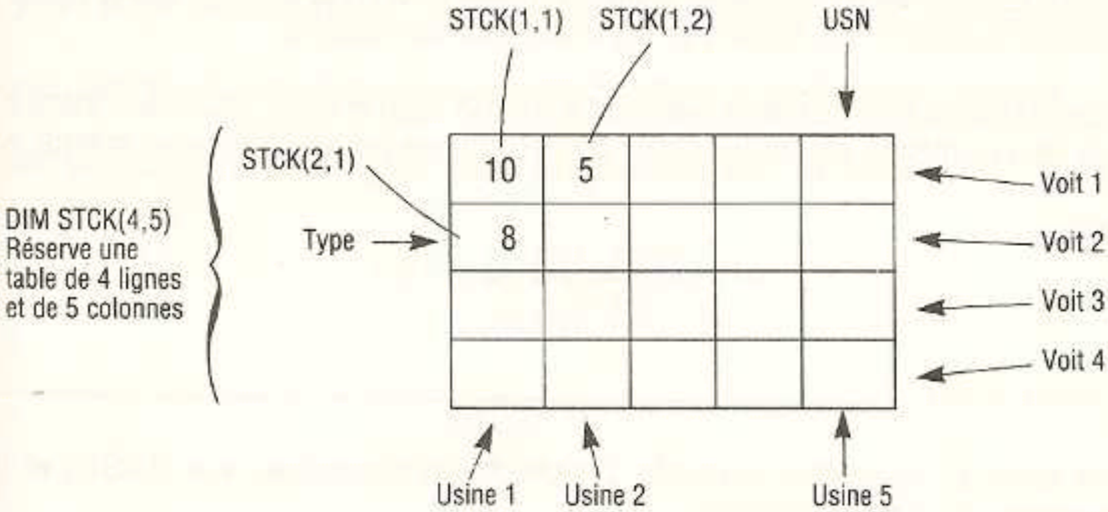
```
100 A(0)=15
```

TABLES A PLUSIEURS DIMENSIONS

Plus généralement, les tables peuvent avoir plusieurs dimensions déclarées par :

DIM nom-table (dim1, dim2, ...)

Soit une table à deux dimensions contenant des stocks de voitures de différents types et de différentes usines.



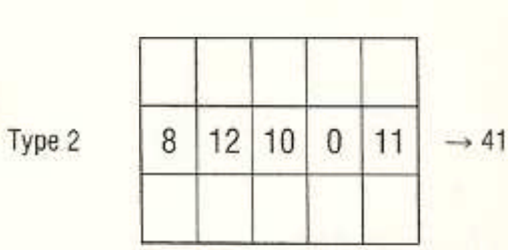
Documentons la table à l'aide de l'instruction INPUT :

```
10 DIM stck(4,5)
20 '
30 FOR type=1 TO 4
40  FOR usn=1 TO 5
50   PRINT "Stock/Type:";type;"Usine:";usn;
60   INPUT stck(type,usn)
70  NEXT usn
80 NEXT type
```

```
Stock/Type: 1 Usine: 1 ? 10
Stock/Type: 1 Usine: 2 ? 5
Stock/Type: 1 Usine: 3 ? 3
Stock/Type: 1 Usine: 4 ? 7
Stock/Type: 1 Usine: 5 ? 9
Stock/Type: 2 Usine: 1 ? 8
Stock/Type: 2 Usine: 2 ?
```

Lorsque la table est documentée, pour connaître le nombre de véhicules d'un type, toutes usines confondues, on fait :

```
100 INPUT "Type ";type
110 '
120 ttal=0
130 FOR usn=1 TO 5
140  ttal=ttal+stck(type,usn)
150 NEXT usn
160 '
170 PRINT "Total type:";type;ttal
```




```
run

saisie table

Type ? 2

Total type: 2   41
```

Comme pour les tables à une dimension, la déclaration d'une table à plusieurs dimensions, toutes inférieures ou égales à 10, n'est pas nécessaire.

Alors qu'une table et une variable peuvent avoir un même nom, plusieurs tables, même si elles ont des nombres de dimensions différents, ne peuvent avoir le même nom.

Exemple :

```
DIM a(7):DIM a(7,8)
```

ERASE nom-table

Efface une table en mémoire centrale. La place est récupérée par BASIC et la table peut à nouveau être dimensionnée.

```
10 DIM a(200)
20 PRINT FRE(0)                    ' Place libre
30 ERASE a                         ' effacement table a()
40 PRINT FRE(0)                    ' Place libre

run
 42377
 43391
```

Lorsqu'une table n'est pas dimensionnée par DIM, elle est dimensionnée par défaut avec 10 éléments par BASIC dès qu'un élément de la table est référencé en lecture ou en écriture.

Si une instruction DIM est exécutée après le dimensionnement par défaut, le message **ARRAY ALREADY DIMENSIONNED** est envoyé par BASIC.

```
10 a(4)=15              ' dimensionnement implicite
20 DIM a(20)
```

Il faut dimensionner la table avant de référencer un élément. Lorsqu'une table n'est pas dimensionnée explicitement par DIM et que l'on essaie de référencer l'élément 11, on obtient le message **SUBSCRIPT OUT OF RANGE**.

```
10 FOR i=1 TO 15
20  PRINT a(i);
30 NEXT i
```

Il faut ajouter :

```
5 DIM A(15)
```

OCCUPATION MÉMOIRE

Par défaut, les tables numériques occupent 5 octets pour chaque élément.

Les tables déclarées entières (%) occupent 2 octets par élément.

```
10 PRINT FRE(0)
20 DIM a(200)
30 PRINT FRE(0)

run
 43494
 42480

10 PRINT FRE(0)
20 DIM a%(100)
30 PRINT FRE(0)

run
 43494
 43283
```

LES CHAÎNES DE CARACTÈRES

- LEFTS
- RIGHT$
- MID$
- LEN
- STR$
- VAL
- ASC
- CHR$
- INSTR
- STRING$
- SPACE$
- HEX$
- BIN$
- UPPER$
- LOWER$

L'affectation d'une valeur à une chaîne s'écrit :

nom de chaîne = « suite de caractères »

Exemple : 10 NOM$ = « DUPONT »

Les « » indiquent que DUPONT doit être interprété comme une chaîne de caractères et non comme une variable. La longueur d'une chaîne de caractères, qui n'a pas à être déclarée, peut varier en cours d'exécution du programme (jusqu'à 255). De même, la longueur de chaque élément d'une table de chaînes peut varier dynamiquement.

La **concaténation** (réunion) de chaînes de caractères est réalisée par l'opérateur noté « + ».

```
10 nom$="DUPONT"
20 pren$="jean"
30 np$=nom$+pren$
40 PRINT np$

run
DUPONTjean
```

COMPARAISONS

La comparaison de chaînes de caractères se fait avec les opérateurs :

=, <, >, >=, <=, <>

Les chaînes sont comparées caractère par caractère de la gauche vers la droite jusqu'à ce que l'un des caractères d'une chaîne soit plus grand que l'autre (code ASCII supérieur). C'est alors cette chaîne qui est considérée comme la plus grande ("DURAND" est plus grand que "DUPONT").
Si tous les caractères sont égaux, les chaînes sont considérées comme égales.

```
10 INPUT "1er nom ";n1$
20 INPUT "2eme nom ";n2$
30 IF n1$>n2$ THEN PRINT n1$;" est plus grand que ";n2$
40 IF n1$<n2$ THEN PRINT n1$;" est plus petit que ";n2$
50 GOTO 10

run
1er nom ? DURAND
2eme nom ? DUPOND
DURAND est plus grand que DUPOND
```

Le programme ci-dessous teste une réponse au clavier.

```
10 INPUT "Reponse (oui ou non) ";r$
20 IF r$="oui" THEN PRINT "Vous avez dit oui"
30 IF r$="non" THEN PRINT "Vous avez dit non"
40 IF r$="" THEN PRINT "Vous avez appuye sur ENTER"
50 GOTO 10
```

Remarquez le test d'une chaîne vide en 40(IF R$=" ").

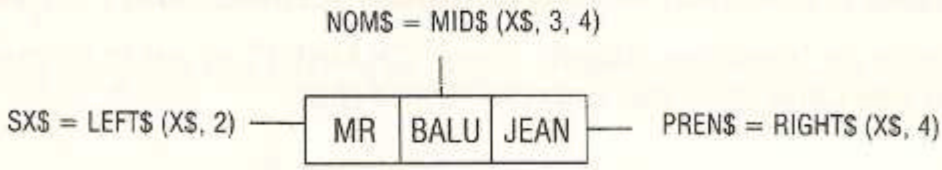
LEFT$,RIGHT$,MID$

LEFT$, RIGHT$, MID$ permettent d'accéder respectivement aux caractères de gauche, de droite et de l'intérieur d'une chaîne.

LEFT$ (CHAINE, longueur à prendre à gauche)
RIGHT$ (CHAINE, longueur à prendre à droite)
MID$ (CHAINE, position début, longueur à prendre)

"**CHAINE**", "**longueur à prendre**" et "**position début**" peuvent être des expressions. Les valeurs de "**longueur à prendre**" et de "**position début**" doivent être comprises entre 0 et 255.

Exemple :




```
10 x$="MRBALUJEAN"
20 sx$=LEFT$(x$,2)            ' 2 caracteres a gauche
30 pren$=RIGHT$(x$,4)         ' 4 caracteres a droite
40 nom$=MID$(x$,3,4)          ' 4 caracteres a partir du 3eme
50 PRINT sx$,pren$,nom$

run
MR             JEAN           BALU
```

Dans l'exemple ci-dessous, nous testons la première lettre de la réponse :

```
10 INPUT "Reponse (OUI/NON) ";r$
20 IF LEFT$(r$,1)="O" THEN GOTO 100
```

Autre exemple :

```
10 nom$="DUBONET"
20 FOR i=1 TO 7
30  PRINT LEFT$(nom$,i)
40 NEXT i

run
D
DU
DUB
DUBO
DUBON
DUBONE
DUBONET
```

Si "**longueur à prendre**" spécifiée est plus grande que la longueur de la chaîne, le résultat est la chaîne elle-même.

Lorsque "**longueur à prendre**" n'est pas précisée dans MID$, cette fonction devient équivalente à RIGHT$.

Si "**position début**" spécifiée dans MID$ est plus grande que la chaîne elle-même, une chaîne vide est retournée.

MID$(CHAINE1, position début, longueur à remplacer) = CHAINE2

Permet, à partir de "**position début**" dans "**CHAINE1**" et sur la longueur spécifiée, de remplacer des caractères par ceux de "**CHAINE2**".

```
10 x$="MRBALUJEAN"
20 MID$(x$,3,4)="XXXX"
30 PRINT x$
40 PRINT nP$

run
MRXXXXJEAN
```

Attention : ne permet pas l'insertion ou la suppression de caractères mais seulement la substitution.

Si "**CHAINE2**" est plus longue que "**longueur à remplacer**", seuls les premiers caractères de "**CHAINE2**" sont pris en considération.

Si "**CHAINE2**" est plus courte que "**longueur à remplacer**", il n'y a substitution des caractères que sur une longueur égale à celle de "**CHAINE2**".

LEN (chaîne)

Donne la longueur d'une expression chaîne.

```
10 nom$="DUPONT"
20 l=LEN(nom$)
30 PRINT l

run
 6
```

STR$(X)

Convertit une expression numérique X en une chaîne de caractères.

```
10 x=123
20 x$=STR$(x)
30 PRINT x$,LEN(x$)

run
 123       4
```

Remarque : Le premier caractère de la chaîne est réservé pour le signe ":" C'est un espace pour un nombre positif et un signe "−" pour un nombre négatif.

```
PRINT MID$(STR$(123),2,1)  --> 1
```

VAL (chaîne)

Fonction inverse de STR$, elle donne la valeur numérique d'une expression chaîne.

```
10 x$="123 francs"
20 x=VAL(x$)
30 PRINT x

run
 123
```

Si le premier caractère n'est pas un caractère décimal, un espace, un signe "+", un signe "−" ou ".", le résultat est égal à zéro.

```
10 PRINT VAL(" 123")
20 PRINT VAL("+123")
30 PRINT VAL("A123")

run
 123
 123
 0
```

ASC (caractère)

Chaque caractère a un code interne (code ASCII) auquel on accède par la fonction ASC (caractère).

```
10 x$="A"
20 x=ASC(x$)
30 PRINT x

run
 65
```

ASC (chaîne)

Donne le code ASCII du premier caractère d'une expression chaîne.

```
10 Print asc("BONJOUR") --> 66
```

Une chaîne nulle comme argument provoque une erreur.

```
10 x$="":PRINT ASC(x$)

Improper argument in 10
```

CHR$(X)

Fonction inverse de la fonction ASC, elle permet de générer des caractères ayant pour code ASCII la valeur de X. Cette valeur doit être comprise entre 0 et 255. X peut être une constante, une variable ou une expression.

```
10 FOR i=65 TO 65+25
20  PRINT CHR$(i);
30 NEXT i

run
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

CHR$(X) est utilisé pour envoyer des "caractères de contrôle" aux périphériques (écran, imprimante).

CHR$(8) provoque un retour arrière du curseur.
CHR$(10) provoque un passage à la ligne (sans retour en début de ligne).
CHR$(13) provoque un retour en début de ligne.

Exemples divers :

- **Suppression d'un caractère à droite d'une chaîne :**

```
10 x$="DUPONT"
20 x$=LEFT$(x$,LEN(x$)-1)
30 PRINT x$

run
DUPON
```

- **Normalisation d'une chaîne à une longueur fixe :**

```
10 x$="DUPONT"
20 y$=RIGHT$("          "+x$,8)
30 PRINT y$,LEN(y$)

run
  DUPONT      8
```

- **Insertion d'un caractère dans une chaîne :**

```
10 z$="AAAAA"
20 x$="B":P=3
30 z$=LEFT$(z$,P)+x$+RIGHT$(z$,LEN(z$)-P)
40 PRINT z$

run
AAABAA
```

- **Remplissage par des zéros à gauche :**

```
10 x=123
20 x$=RIGHT$(STR$(100000+x),5)
30 PRINT x$

run
00123
```

INSTR (position départ, chaîne, chaîne cherchée)

Recherche la position d'une chaîne dans une autre. Par défaut, la position de départ est égale à 1 :

- Si la chaîne cherchée n'est pas trouvée, le résultat est égal à 0.
- Si la chaîne cherchée est nulle, le résultat est la position de départ spécifiée.
- Si la position de départ est supérieure à la longueur de la chaîne où s'effectue la recherche, le résultat est nul.

```
10 x$="DUPONT.JEAN"
20 P=INSTR(x$,".")
30 y$=LEFT$(x$,P-1)
40 PRINT y$,P

run
DUPONT        7
```

```
10 PRINT INSTR(3,"ABAABAAA","B")  --> 5
```

Le programme ci-dessous vérifie si un nom appartient à un ensemble.

```
10 e$="JEAN PIERRE PAUL JACQUES"
20 INPUT "Nom ";nom$
30 P=INSTR(e$,nom$):IF P=0 THEN PRINT "Erreur":GOTO 20
```

Cette séquence d'instructions permet de répondre à une question "Mode ?" non pas par un chiffre, mais par une lettre (plus mnémonique).

```
10 INPUT "Mode (C,A,P,R,N) ";m$          ' entrer C,A,P,..
15 '
20 P=INSTR(" CAPRMN",m$)                 ' Position caractere frappe
30 IF P<2 THEN 10                        ' validite?
40 ON P-1 GOTO 100,200,400,500,600
50 '
100 PRINT "ligne 100":STOP
200 PRINT "ligne 200":STOP
400 PRINT "ligne 400":STOP
```

Remarque : Si M$ est une chaîne vide, P est égal à 1. C'est ce qui explique la présence d'un espace dans l'instruction 20 devant "CAPRMN"

La séquence d'instructions ci-dessous vérifie si un caractère frappé appartient bien à un ensemble de caractères autorisés : (chiffres plus le "." sur l'exemple).

```
10 c$=INKEY$:IF c$="" THEN 10      ' attente caractere
20 P=INSTR("0123456789.",c$)       ' Position caractere frappe?
30 IF ASC(c$)=13 THEN 90           ' ENTER?
40 IF P=0 THEN PRINT CHR$(7);:GOTO 10 ' caractere invalide?
50 lig$=lig$+c$
60 PRINT c$;
70 GOTO 10
80 '
90 PRINT:PRINT lig$
```

STRING$ (nombre de fois, chaîne)

Génère une chaîne de caractères égale à la chaîne spécifiée, multipliée par le nombre de fois indiqué.

```
10 x$=STRING$(10,".")            ' chaine de 10 '.'
20 PRINT x$

run
..........
```

SPACE$(X)

Génère une chaîne de X espaces. X peut être une expression.

```
10 x$="DUPONT"+SPACE$(5)+"JEAN"
20 PRINT x$

run
DUPONT     JEAN
```

Notons que la fonction SPC ne génère pas une chaîne d'espaces et n'est utilisable qu'avec l'instruction PRINT.

HEX$(expression)

Fournit une chaîne représentant la valeur hexadécimale de la valeur donnée. La valeur donnée doit être entière (−32768 à +65535).

```
10 X$=HEX$(14):Y$=HEX$(14,2)
20 PRINT X$,Y$

RUN
E             0E
```

BIN$(expression)

Fournit une chaîne représentant la valeur binaire de la valeur donnée.

```
10 X$=BIN$(12):Y$=BIN$(12,8)
20 PRINT X$,Y$
RUN
1100          00001100
```

UPPER$(chaîne)

Convertit une chaîne en majuscules.

```
10 nom$="roulet"
20 PRINT UPPER$(nom$)

run
ROULET
```

LOWER$(chaîne)

Convertit une chaîne en minuscules.

```
10 nom$="ROULET"
20 PRINT LOWER$(nom$)

run
roulet
```

Remarques sur les chaînes :

Lorsqu'une chaîne voit sa longueur changée, elle est déplacée.

La place occupée par l'ancienne chaîne reste perdue jusqu'à ce que l'espace mémoire soit réorganisé par BASIC. Pour éviter ces réorganisations (longues), on pourra utiliser **MID$(x$,p,1)="XX"** qui modifie la chaîne sans la déplacer.

Le programme ci-dessous remplit une table de chaînes.

PRINT FRE(" ") provoque le tassement des chaînes par BASIC.

```
10 DIM a$(500)
20 hd=TIME
30 FOR i=1 TO 500
40  a$(i)="BASIC"+STR$(i)
50 NEXT i
60 PRINT "Temps:";(TIME-hd)/300
70 hd=TIME
80 PRINT "Espace libre:";FRE(0)
90 PRINT FRE("")                    ' reorganisation chaines
100 PRINT "Temps:";(TIME-hd)/300
110 PRINT "Espace libre:";FRE(0)

run
Temps: 5.05666667
Espace libre: 35441
 37333
Temps: 28.2666667

Espace libre: 37333
```

LES ÉDITIONS

- PRINT
- PRINT,
- PRINT;
- PRINT TAB
- PRINT SPC
- PRINT USING
- POS/VPOS
- WRITE
- ZONE
- WIDTH

PRINT expression

Un simple PRINT d'une constante, variable ou expression, affiche la valeur de celle-ci puis provoque un retour en début de ligné et un saut de ligne.

```
10 x=123
20 PRINT x

run
 123
```

PRINT,

L'impression de plusieurs valeurs sur une même ligne peut se faire simplement en séparant dans l'instruction les noms des variables ou les expressions par des virgules.

```
10 somme=200:nombre=10
20 PRINT somme,nombre,somme/nombre

run
 200          10           20
```

Mais dans ce cas, l'impression des valeurs est faite selon un format standard (colonnes 1, 14, 27, etc., les valeurs sont cadrées à gauche). Zone permet de changer le format standard.

PRINT;

Un point-virgule en fin d'instruction PRINT empêche le saut de ligne : les chaînes de caractères sont concaténées, les valeurs numériques sont suivies par un espace et précédées soit par un espace pour les nombres positifs, soit par le signe "–" pour les nombres négatifs.

```
10 nom$="DUPONT"
20 Pren$="Jean"
30 PRINT nom$;
40 PRINT Pren$

run
DUPONTJean
```

```
10 Print 123;456;-789          -->  123  456 -789
```

PRINT TAB(X)

La fonction TAB(X) permet de positionner directement le curseur à l'intérieur d'une ligne en colonne X, X peut être une expression.

```
10 a=123:b=456
20 PRINT a TAB(15) b

run
 123           456
```

Attention : Le curseur ne peut revenir en arrière. Si X spécifié est plus petit que la position courante du curseur, l'édition se fait sur la ligne suivante.

PRINT SPC(X)

Avec la fonction SPC(X), X blancs sont imprimés à partir de la position courante :

```
10 nom$="DUPONT":Pr$="JEAN"
20 PRINT Pr$;SPC(3);nom$

run
JEAN   DUPONT
```

PRINT USING

Considérons maintenant l'outil d'édition le plus puissant du BASIC : le "PRINT USING"

■ **VARIABLES NUMÉRIQUES : sans** le **PRINT USING** les **valeurs numériques** sont **cadrées** à **gauche**. Or, c'est généralement à **droite** qu'elles doivent être **cadrées.**

■ **PRINT USING « # # # # # » ; expression numérique**

Un format défini par une chaîne de # nous permet de cadrer les nombres à droite. Chaque # représente la position d'un chiffre.

```
10 x=123:y=1234
20 PRINT USING " #### ";x        <-- Format
30 PRINT USING " #### ";y

run
  123
 1234   <-- Cadrage à droite
```

Sans le PRINT USING, nous aurions obtenu des chiffres cadrés à gauche.

```
123
1234
```

La partie entre guillemets qui représente le format peut aussi être définie par une variable chaîne de caractères :

```
10 x=123:y=1234
20 fmt$="####"
30 PRINT USING fmt$;x
40 PRINT USING fmt$;y

run
  123
 1234
```

■ **PRINT USING** « # # # #.# # » **; expression numérique :**

Le nombre de chiffres après la virgule qui doivent être imprimés est précisé dans le format par le nombre de # après le "."

```
10 x=123.456
20 PRINT USING "####.##";x

run
 123.46
```

2 chiffres après la virgule

On remarque que l'arrondi est assuré automatiquement.

■ **PRINT USING «LIBELLÉ** # # # #.# # **», variable numérique**

Un libellé peut être inséré dans le format.

```
10 somme=123.456
20 PRINT USING " Total: ####.##  FRANCS ";somme

run
 Total:  123.46  FRANCS
```

Formats multiples :

Plusieurs formats peuvent être spécifiés dans une seule instruction.

```
10 somme=1234.567:tva=17.6
20 PRINT USING "Total: ####.##  Tva ##.## ";somme,tva

run
Total: 1234.57  Tva 17.60
```

Si le format est le **même** pour plusieurs variables, on fait :

```
10 x=123:y=456
20 PRINT USING "####.##";x,y

run
 123.00 456.00
```

(+)

Le signe "+" n'est imprimé que s'il est prévu dans le programme.

```
10 Print using " ####";123    -->  123
10 Print using "+####";123    --> +123
10 Print using "+####";-123   --> -123
```

(−)

Un signe "−" en fin de format provoque l'impression du signe "−" à la fin d'un nombre négatif.

```
10 Print using "####-";-123   -->  123-
10 Print using "####-";123   -->   123
```

(**)

2 "*" placés en tête de format provoquent le remplissage par des "*" des positions inoccupées à gauche du nombre imprimé. En outre ces 2 "*" spécifient des positions pour 2 chiffres supplémentaires.

```
10 Print using "**####";123     -->  ***123
10 Print using "**####";1234    -->  **1234
```

($$)

**2 "$"** en tête de format provoquent l'impression du caractère "$" à gauche du nombre imprimé.

```
10 Print using "$$####";123   --> $123
```

(**$)

"$"** combine les effets "**" et "$$".

```
10 Print using "**$####";123   --> ***$123
```

(%)

Si le nombre de positions spécifié dans le format est insuffisant pour la valeur à imprimer, **le message %** est imprimé devant la valeur.

```
10 Print using "###";1234   --> %1234
```

■ CHAÎNES DE CARACTÈRES

PRINT USING «\   \» ; expression chaîne :

Le nombre d'espaces entre les "\" définit le nombre de caractères à imprimer−2.

```
10 x$="RENAULT"
20 PRINT USING "\   \";x$

run
RENAU
```

3 espaces
3 + 2 caractères

PRINT USING « ! » ; expression chaîne :

Imprime le premier caractère d'une expression chaîne.

PRINT « & » ; expression chaîne :

Spécifie une longueur variable de chaîne. La chaîne imprimée est égale à la chaîne spécifiée.

```
10 nom$="ROULET":PREN$="NICOLAS"
20 PRINT USING "& !";nom$,Pren$

run
ROULET N
```

Programme pour tester PRINT USING :

Le programme suivant permet d'entrer par INPUT à la fois le format et le nombre à imprimer.

```
10 INPUT "Format,nombre ";fmt$,nombre
20 PRINT USING fmt$;nombre

run
Format,nombre? ####,123
 123
```

POS/VPOS

Donnent les coordonnées du curseur.

```
10 LOCATE 10,8
20 PRINT "NICOLAS";
30 PRINT POS(#0),VPOS(#0)

run
          NICOLAS   17   8
```

WRITE expression1, expression2

Procède comme PRINT ... mais sépare les valeurs par des virgules et imprime les chaînes entre guillemets.

```
10 a=80:b=100:x$="NICOLAS"
20 WRITE a,b,x$

run
80,100,"NICOLAS"
```

ZONE intervalle

Définit l'intervalle d'édition standard lorsque les variables sont séparées par des virgules.

```
10 ZONE 13
20 PRINT 1,2,3
30 ZONE 10
40 PRINT 1,2,3

run
 1            2            3
 1         2         3
```

WIDTH largeur

Définit la largeur d'édition à l'imprimante.

ÉDITIONS IMPRIMANTE :

Pour éditer sur imprimante, il faut utiliser le canal #8.

```
10 PRINT #8,"DUPONT

BLABLA
```

ÉDITIONS IMPRIMANTE/ÉCRAN :

Un même programme peut aiguiller des résultats vers l'écran ou l'imprimante.

```
10 INPUT "Ecran ou imprimante (E/I) ";m$
20 '
30 IF m$="E" THEN cn=0
40 IF m$="I" THEN cn=8
50 '
60 PRINT #cn,"BLABLA"
70 GOTO 10
```

DÉCOUPAGE DES PROGRAMMES | 4

LES SOUS-PROGRAMMES

■ **GOSUB... RETURN**

GOSUB RETURN

Il est fréquent qu'une même séquence d'instructions soit utilisée **plusieurs fois** dans un programme. Un sous-programme permet d'écrire **une seule fois** cette séquence qu'il suffit d'appeler de différents endroits du programme par GOSUB n° d'instruction.

```
10 a=30:b=10
15 '
20 s=a+b
30 p=a*b
50 PRINT a,b,"Somme:";s;"Produit:";p
60 '
70 '
90 a=20:b=5
100 s=a+b
110 p=a*b
120 PRINT a,b,"Somme:";s;"Produit:";p
130 '
```

Même séquence d'instructions

20 GOSUB 500 provoque un branchement du programme en 500 (comme le ferait GOTO 500) mais l'instruction RETURN (RETOUR)) placée à la fin du sous-programme provoque un **retour automatique** après l'instruction qui suit GOSUB 500, c'est-à-dire l'instruction 30 sur l'exemple.

Pour le deuxième appel du sous-programme en 100, le retour se fait en 110.

```
10 a=30:b=10
20 GOSUB 500
30 '
40 '
90 a=20:b=5
100 GOSUB 500
110 END
130 '
500 PRINT "Debut sous-Programme
510 s=a+b
520 p=a*b
530 PRINT a;b;"Somme";s;"Produit";p
540 PRINT "Fin sous-Programme"
550 RETURN
```

```
500 PRINT "DEBUT SOUS-PROGRAMME"
510 S=A+B
520 P=A*B
530 PRINT A;B;"SOMME";S;"PRODUIT";P
540 PRINT "FIN SOUS-PROGRAMME"
550 RETURN
```

On remplace par un sous-programme

Retour automatique à l'instruction qui suit GOSUB

```
run
Debut sous-Programme
 30  10 Somme 40 Produit 300
Fin sous-Programme
Debut sous-Programme
 20  5 Somme 25 Produit 100
Fin sous-Programme
```

Remarque : Un sous-programme peut lui-même en appeler un autre.

Les adresses de retour (90 et 540 sur l'exemple) sont gérées par Basic à l'aide d'une pile.

50	500	1000 .....
60	510	1010 .....
70		
80 GOSUB 500	530 GOSUB 1000	1030 .....
90	540	1040 .....
100	550	1050 .....
110	560 RETURN	1060 RETURN

Pile des adresses de retour
540
90

RETURN

Les instructions sont exécutées dans l'ordre suivant :

50 60 70 80	
500 510 520 530	Début du 1er sous-programme
1000 1010 ... 1050 1060	2e sous-programme
540 550 560	Fin du 1er sous-programme
90 100	Retour au programme principal

N'essayez pas de sortir d'un sous-programme par GOTO, ni d'entrer dans un sous-programme par GOTO. Les sous-programmes sont utilisés non pas seulement pour économiser de la place mémoire, mais aussi (et plutôt) pour les raisons suivantes :

- Par souci de clarté : le "programme principal" est plus court.
- Les sous-programmes permettent éventuellement de répartir la programmation entre plusieurs programmeurs.
- La mise au point se fait sous-programme par sous-programme. Ainsi on progresse plus sûrement dans la mise au point de l'ensemble du programme.
- Dans le cas où la même séquence d'instructions est répétée, une modification dans cette séquence doit être faite aux différents endroits où elle a été écrite (fastidieux).

- Lorsqu'un sous-programme est modifié, la "zone d'intervention" est bien délimitée ; tout le programme ne risque pas d'être remis en cause.
- Les sous-programmes ont aussi l'avantage, lorsqu'en cours de mise au point, il faut "restructurer" le programme, de permettre une certaine souplesse ; l'ordre d'appel des sous-programmes est à modifier mais ceux-ci ne changent pas.
- Plutôt que d'insérer une nouvelle séquence d'instructions correspondant à l'ajout d'une nouvelle fonction, on peut écrire un sous-programme qui est appelé quand nécessaire.

Les avantages des sous-programmes sont donc multiples. Aussi n'hésitera-t-on pas à en faire un usage abondant, même lorsqu'ils ne comportent que quelques instructions.

LES BRANCHEMENTS MULTIDIRECTIONS

- **ON X GOTO...**
- **ON X GOSUB...**

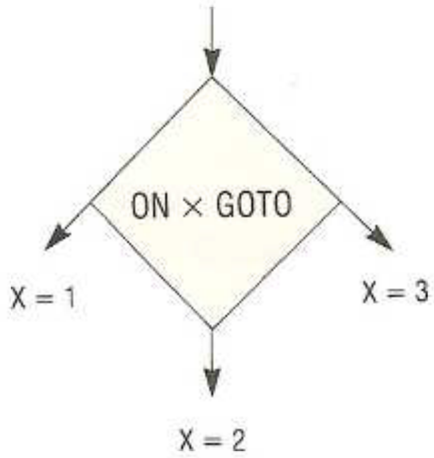
ON X GOTO n° ligne1, n° ligne2

Suivant la valeur d'une variable (ou d'une expression) : 1, 2, 3,... il y a **branchement** au n° de ligne1, n° de ligne2, n° de ligne3,...

S'il ne correspond pas de numéro de ligne à la valeur de la variable, c'est l'instruction qui suit "ON GOTO" qui est exécutée.

```
10 INPUT "Votre choix (1,2,3) ";ch
20 '
30 ON ch GOSUB 100,300,500
40 '
50 STOP          CH = 1  CH = 2  CH = 3
60 '
100 PRINT "Choix 1"
120 STOP
130 '
300 PRINT "Choix 2"
310 STOP
330 '
500 PRINT "Choix 3"
510 STOP

run
Votre choix (1,2,3) ? 2
Choix 2
```



ON X GOSUB n° ligne1, n° ligne2,...

Fonctionne comme ON X GOTO, mais dès qu'une instruction RETURN est rencontrée, il y a retour automatique après l'instruction ON X GOSUB...

```
10 INPUT "Votre choix (1,2,3) ";ch
20 '
30 ON ch GOSUB 100,300,500
40 PRINT "Suite"
50 GOTO 10
60 '
100 PRINT "Choix 1"
120 RETURN
130 '
300 PRINT "Choix 2"
310 RETURN
330 '
500 PRINT "Choix 3"
510 RETURN
```

LE TRAITEMENT D'ERREURS | 5

- ERR
- ERL
- ON ERROR GOTO
- RESUME
- ERROR

ERR-ERL

Dès que survient une erreur pendant l'exécution d'un programme et si l'instruction "ON ERROR GOTO n° ligne" a été prévue, il y a branchement à un programme d'erreur au numéro de ligne spécifié dans l'instruction "ON ERROR".

Ce programme d'erreur analyse alors l'erreur en testant les valeurs de **ERR** et **ERL** qui représentent respectivement le code erreur et le numéro de ligne où s'est produite l'erreur.

Après avoir analysé et traité l'erreur, l'instruction RESUME permet au programme d'erreur de provoquer un retour au programme où s'était produite l'erreur.

```
10 ON ERROR GOTO 100                          en cas d'erreur
15 '
20 INPUT "Diviseur";d
30 PRINT 10/d
40 GOTO 20
90 '------------------------- analyse erreur
100 PRINT "ERR=";ERR;"ERL=";ERL
110 IF ERR=11 AND ERL=30 THEN PRINT "Division Par zero inter
dite":RESUME 20
120 PRINT "Erreur non reconnue":STOP

run
Diviseur? 5
 2
Diviseur? 0
Division Par zero interdite
Diviseur?
```

ON ERROR GOTO 0

Ecrit dans un programme de traitement d'erreur, il annule ON ERROR GOTO N° ligne. L'erreur est donc traitée normalement par le système (interruption du programme et message d'erreur).

Remarque : la division par zéro n'arrête pas l'exécution du programme.

RESUME

Ecrit à la fin d'un programme de traitement d'erreur, RESUME spécifie où doit se poursuivre l'exécution du programme :

RESUME	l'exécution se poursuit au numéro de ligne où s'est produite l'erreur.
RESUME NEXT	l'exécution se poursuit au numéro de ligne après celui où s'est produite l'erreur.
RESUME N° ligne	l'exécution se poursuit au numéro de ligne spécifié.

ERROR

ERROR N° erreur	permet à l'utilisateur de définir ses propres codes erreurs (compris entre 0 et 255) et de provoquer un branchement à ON ERROR GOTO... comme si une erreur avait eu lieu.
10 ERROR 11	provoque un branchement à ON ERROR GOTO...

Remarque : "ON ERROR GOTO N° ligne" doit être écrit de préférence en tête de programme afin d'être interprétée avant que ne survienne une erreur.

LA MISE AU POINT DES PROGRAMMES | 6

- ESC
- CONT
- STOP/END
- TRON
- TROFF

Les programmes ne fonctionnent pas toujours “du premier coup”. BASIC envoie des messages d'erreurs (de syntaxe par exemple) mais ne détecte pas les erreurs de logique. Pour les cas les plus délicats, il faut suivre le déroulement du programme étape par étape, ce qui est relativement simple en BASIC.

ESC - CONT

En appuyant sur ESC deux fois (une fois pour l'instruction INPUT) nous interrompons l'exécution du programme. Nous pouvons alors visualiser les valeurs des variables en mode immédiat.

L'exécution interrompue peut être poursuivie en frappant CONT (continue).

Exemple :

Le programme ci-dessous effectue la moyenne de plusieurs notes. Nous avons commis (volontairement !) une erreur. En 50, au lieu de TNTE = TNTE + NTE, nous avons écrit TNTE = NTE.

```
10 tnte=0
15 nnte=0
20 '
30 INPUT "NOTE ";nte
40 IF nte=0 THEN 100
50 tnte=nte  <-- Erreur! TNTE = TNTE + NTE
60 nnte=nnte+1
70 GOTO 30
80 '
100 PRINT "Moyenne:";tnte/nnte
```

```
run
NOTE ? 15
NOTE ? 13
NOTE ? 18
NOTE ? 0     <-- On entre 0 pour indiquer la fin des notes
Moyenne: 6   <-- La moyenne devrait être 15,3
```

Remarque : Les instructions 10 et 15 qui initialisent les valeurs de TNTE et NNTE à zéro ne sont pas indispensables puisque RUN les initialise à zéro. Il est cependant plus prudent de le faire dans un programme plus important où TNTE et NNTE pourraient déjà avoir été utilisées dans une autre partie de programme et avoir une valeur non nulle.

Exécutons à nouveau le programme et interrompons-le après avoir entré la première note. Nous pouvons visualiser en mode direct les valeurs des variables NTE, TNTE et NNTE.

```
run
NOTE ? *Break*
Break in 30

PRINT NTE,TNTE,NNTE
15            15            1
```

On appuie sur ESC pour interrompre

Visualisation des valeurs en mode direct

Pour l'instant, rien d'anormal.

Frappons CONT pour continuer l'exécution du programme et interrompons à nouveau le programme après avoir entré la deuxième note :

```
CONT
NOTE ? 13
NOTE ? *Break*
Break in 30

PRINT NTE,TNTE,NNTE
13            13            2
```

On appuie à nouveau sur ESC

Le cumul pour TNTE n'est pas effectué

Nous nous apercevons en regardant la valeur de TNTE que le cumul des notes n'est pas effectué.

STOP

Au lieu d'appuyer sur la touche ESC, nous aurions pu placer une instruction "STOP" en 65.

Ajouter cette instruction →

```
65 STOP
run
NOTE ? 15

Break in 65
Ready
Print tnte,nnte
 15            1
```

On peut voir à chaque passage dans la boucle comment évoluent les valeurs des variables TNTE et NNTE

```
cont
NOTE ? 13

Break in 65

Print tnte,nnte
 13            2
```

En cas de "BOUCLAGE" de programme :

Lorsqu'un programme ne s'arrête pas, appuyer sur ESC ; frapper TRON puis CONT. On peut ainsi localiser la partie du programme où la boucle s'effectue.

TRON-TROFF

Les commandes "TRON" et "TROFF" peuvent être incluses comme instructions dans un programme.

Exemple :

```
10 i=1
20 '
30 i=i+1
40 IF i=5 THEN 70
50 GOTO 40
60 '
70 END

tron

run
[10][20][30][40][50][40][50][40][50][40][50][40][50]
[40][50][40]

troff
```

programme

LES FONCTIONS | 7

LES FONCTIONS ARITHMÉTIQUES

L'argument des fonctions trigonométriques est normalement exprimé en radians. L'instruction DEG permet de l'exprimer en degrés. Le retour aux radians se spécifie par l'instruction RAD.

```
10 RAD
20 PRINT SIN(PI/2)
30 DEG
40 PRINT SIN(90)
50 RAD

run
 1
 1
```

ABS(X) Fournit la valeur absolue de X :

100 PRINT ABS(−35) ⟶ 35

ATN(X) Donne en radians l'arctangeante de X.

CINT(X) Convertit X en un entier avec arrondi :

100 PRINT CINT(1.6) ⟶ 2
110 PRINT CINT(−1.2) ⟶ −1

COS(X) Donne le cosinus de X.

CREAL(X) Convertit un nombre en réel.

EXP(X) Donne l'exponentielle de X.

FIX(X) Supprime les chiffres après la virgule :

100 PRINT FIX(2.2) ⟶ 2
110 PRINT FIX(−2.2) ⟶ −2

FRE(0) Donne la place libre en mémoire centrale (l'argument 0 n'est pas utilisé).

INT(X) Donne la partie entière de X avec arrondi :

```
100 PRINT INT( 2.2)  ——→  2
110 PRINT INT(-2.2)  ——→  -3
```

LOG(X) Fournit le logarithme de X.

LOG10(X) Fournit le logarithme en base 10.

```
100 PRINT LOG10(100)  ——→  2
```

MAX(X,..,Z) Donne le maximum d'une liste.

```
100 PRINT MAX(15,4)  ——→  15
```

MIN(X,..,Z) Donne le minimum d'une liste.

```
100 PRINT MIN(15,4)  ——→  4
```

ROUND(X,N) Donne l'arrondi de X avec N décimales.

```
100 PRINT ROUND (123.456,2)  ——→  123.46
```

SGN(X) Donne le signe X : on obtient respectivement −1, 0, +1 pour les valeurs négatives, nulles, positives.

SIN(X) Donne le sinus de X.

SQR(X) Donne la racine carrée de X (X doit être positif).

TAN(X) Donne la tangente de X.

UNT(X) Convertit un entier sans signe en un entier entre −32768 et +32767.

DEF FN(X,Y,Z)

En plus des fonctions internes (telles que SQR, SGN, INT, etc.), l'utilisateur peut définir ses propres fonctions par :

```
DEF FNXX (X,Y,Z,...) = EXPRESSION (X,Y,Z,...)
```

où XX représente un nom choisi par l'utilisateur pour identifier sa fonction (les règles d'appellation sont les mêmes que pour les variables) et X,Y,Z,... les arguments de la fonction.

Plus tard, cette fonction sera appelée par le programme avec les valeurs réelles des paramètres.

L'exemple ci-dessous donne la moyenne de trois nombres.

```
10 DEF FNmoy(x,y,z)=(x+y+z)/3
20 '
30 PRINT FNmoy(14,6,10)

run
 10
```

Une fonction ne peut être écrite que sur une seule ligne de 255 caractères au plus. C'est de préférence en tête de programme que sont écrites les fonctions de façon à être interprétées avant qu'elles ne soient appelées.

programme

L'ACCÈS À LA MÉMOIRE ET ENTRÉES-SORTIES DIRECTES | 8

L'ACCÈS À LA MÉMOIRE

- PEEK
- POKE
- HIMEM
- MEMORY
- INP
- OUT
- WAIT

PEEK (adresse mémoire)

Fournit en décimal le contenu d'un octet de la mémoire.

```
10 FOR m=1 TO 5
20  PRINT m,PEEK(m)
30 NEXT m

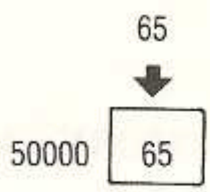
 1              137
 2              127
 3              237
 4              73
 5              195
```

POKE adresse mémoire, valeur

Range une valeur exprimée en décimal (0→255) à l'adresse spécifiée.

```
10 POKE 50000,65     ' 65 dans 50000
20 PRINT PEEK(50000)' affiche le contenu de 50000

run
 65
```



Naturellement, il faut que la mémoire adressée soit modifiable.

Le programme ci-dessous visualise le contenu de la mémoire à l'écran ou sur imprimante. Les adresses mémoires et les contenus sont spécifiés en hexadécimal. Pour les visualiser en décimal, supprimer HEX$.

```
10 '--------------------- dump memoire
20 MODE 2
30 INPUT "Adresse depart (en decimal ou hexa  (&xxx)) ";ad
40 INPUT "Combien d'octets ";n
50 INPUT "Imprimante (O/N) ";r$
60 IF r$="O" OR r$="o" THEN p=8 ELSE p=0
70 '
80 FOR am=ad TO  ad+n STEP 8
90  PRINT #p,"&";HEX$(am);TAB(8);
100  lg$=""
110  FOR k=0 TO 7
120   c=PEEK(am+k)
130   PRINT #p,HEX$(c);TAB(12+k*4);
140   IF c>31 AND c<128 THEN lg$=lg$+CHR$(c) ELSE lg$=lg$+CHR$(32)
150  NEXT k
160  PRINT #p,lg$
170 NEXT am
```

```
Adresse depart (en decimal ou hexa  (&xxx)) ? &168
Combien d'octets ? 200
Imprimante (O/N) ?
```

```
&168   0   0   0   0   0   0   0   0
&170   2A  0   A   0   1   C0  2D  2D   *     --
&178   2D  2D  2D  2D  2D  2D  2D  2D   --------
&180   2D  2D  2D  2D  2D  2D  2D  2D   --------
&188   2D  2D  2D  2D  20  64  75  6D   ---- dum
&190   70  20  6D  65  6D  6F  69  72   p memoir
&198   65  0   8   0   14  0   AD  20   e
&1A0   10  0   3B  0   1E  0   A3  20     ;
&1A8   22  41  64  72  65  73  73  65   "Adresse
&1B0   20  64  65  70  61  72  74  20    depart
&1B8   28  65  6E  20  64  65  63  69   (en deci
&1C0   6D  61  6C  20  6F  75  20  68   mal ou h
&1C8   65  78  61  20  20  28  26  78   exa  (&x
&1D0   78  78  29  29  20  22  3B  D    xx)) ";'
&1D8   6   0   61  E4  0   1F  0   28     a    (
&1E0   0   A3  20  22  43  6F  6D  62      "Comb
&1E8   69  65  6E  20  64  27  6F  63   ien d'oc
```

HIMEM

Fournit l'adresse mémoire maxi utilisée par BASIC.

```
10 PRINT HIMEM
```

MEMORY adresse

Modifie l'adresse mémoire la plus haute de la mémoire pour BASIC.

```
10 PRINT HIMEM
20 MEMORY 40000
30 PRINT HIMEM

run
43143
40000
```

INP (n° entrée)

Lit sur l'entrée spécifiée une valeur comprise entre 0 et 255.

```
10 PRINT INP(&FF177)
```

OUT n° sortie, valeur

Envoie vers le numéro de sortie indiqué la valeur spécifiée (entre 0 et 255).

```
OUT %F8F4,10
```

WAIT n° entrée, octet

Suspend l'exécution du programme jusqu'à ce que (valeur entrée) AND (octet) <> 0.

```
WAIT &FF34,20
```

programme

LES NOMBRES ALÉATOIRES | 9

- **RND**
- **RANDOMIZE**

Les nombres aléatoires sont essentiellement utilisés pour les programmes de jeu ou d'éducation.

RND(X) pour X > 0 ou RND

Fournit un nombre aléatoire compris entre 0 et 1 (1 exclus).

```
10 FOR n=1 TO 3
20  PRINT RND(1),
30 NEXT n

run
 0.271940658            0.528612386            0.021330127

run
 0.175138616            0.657773343            0.653729687
```

Pour obtenir des nombres entiers entre 0 et 9 par exemple, il faut :

- □ multiplier le nombre obtenu par 10,
- □ prendre la partie entière du résultat, avec la fonction INT (X).

```
10 FOR n=1 TO 4
20  x=RND(1)
30  y=x*10
40  z=INT(y)
50  PRINT x,y,z
60 NEXT

 0.810653739            8.10653738    8
 0.173633121            1.73633121    1
 0.350786642            3.50786643    3
 0.580428399            5.80428399    5
```

Pour obtenir un nombre entier entre 1 et 10, il suffit d'ajouter 1.

```
10 FOR n=1 TO 6
20  PRINT INT(RND*10)+1;
30 NEXT n

 3  8  8  1  4  2
```

RND(0)

Fournit le dernier nombre aléatoire généré.

```
10 FOR n=1 TO 3
20  PRINT RND
30 NEXT n
40 PRINT RND(0)

 0.182864718
 0.352758596
 0.375612436
 0.375612436
```

RANDOMIZE

Initialise une série aléatoire qui dépend de X. La série est toujours la même pour X donné.

```
10 RANDOMIZE 6              ' initialise une serie
20 '
30 FOR i=1 TO 3
40  PRINT RND(1)
50 NEXT i

run
 0.271940658
 0.528612386
 0.021330127

run
 0.271940658
 0.528612386
 0.021330127
```

Pour générer des séries aléatoires différentes, faire :

RANDOMIZE TIME.

```
10 RANDOMIZE TIME
20 FOR x=1 TO 3
30  PRINT, RND(1)
40 NEXT x
 0.369625518
 0.925758611
 0.766109714

 0.782020996
 1.43015E-02
 0.847328433
```

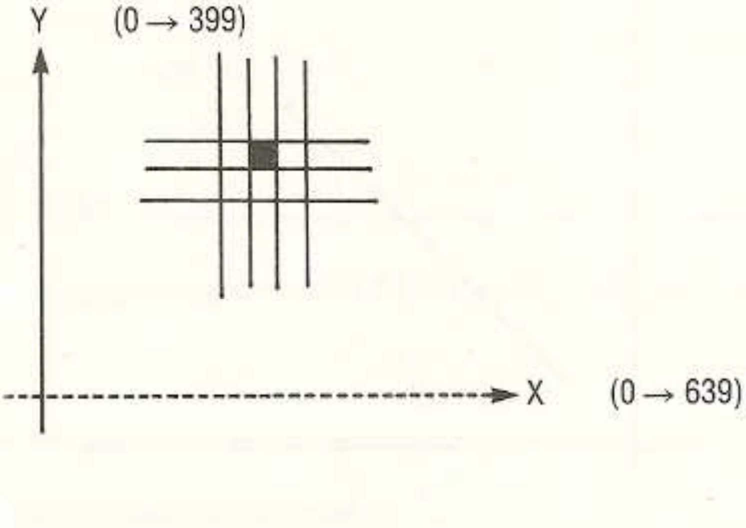
GRAPHISMES ET SONS | 10

LE GRAPHISME HAUTE RÉSOLUTION

- MODE
- PLOT/PLOTR
- DRAW/DRAWR
- MOVE/MOVER
- XPOS/YPOS
- CLG
- TEST/TESTR
- TAG/TAGOFF
- ORIGIN

MODE

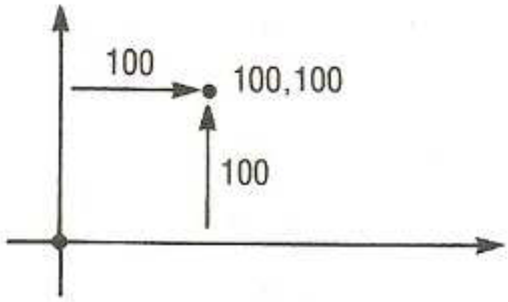
L'écran est divisé en 400*640 points. Il existe trois modes : **MODE 0, MODE 1, MODE 2**. Suivant le mode, la taille des points allumés est plus ou moins importante, mais l'adressage se fait **toujours** dans la résolution la plus élevée, c'est-à-dire 400*640. L'adressage se fait soit en absolu, soit en relatif.



PLOT X,Y,stylo
PLOTR DX,DY,stylo

PLOT X,Y allume le point X,Y spécifié.

```
10 MODE 2
20 PLOT 100,100,1
```



Allume le point de coordonnées 100,100.

Si le mode spécifié est MODE 1, le point allumé est plus gros.

```
10 MODE 1
20 PLOT 100,100,1
```

Mais l'affichage se fait au même endroit.

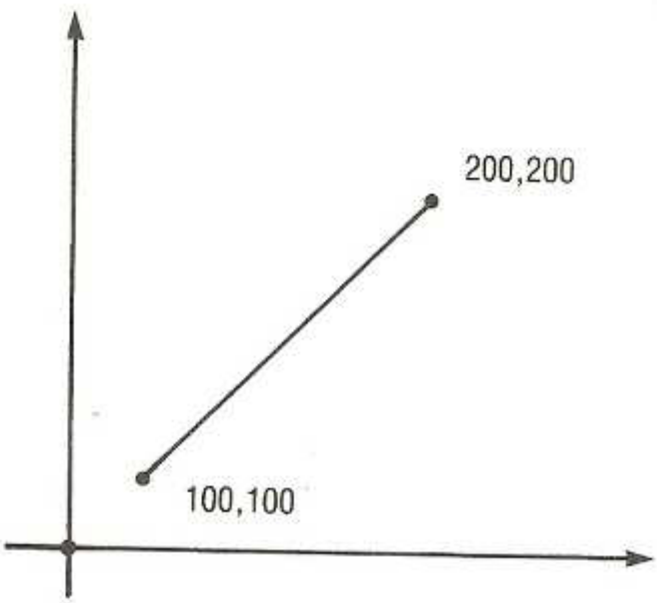
PLOTR DX,DY allume un point par rapport au point courant :

```
10 MODE 2
20 PLOT 100,100,1
30 PLOTR 20,20,1          ' decalage de 20,20 (120,120 en absolu)
```

DRAW X,Y,stylo
DRAWR DX,DY,stylo

DRAW trace une droite entre le point courant et le point spécifié.

```
10 MODE 2
20 PLOT 100,100
30 DRAW 200,200          ' droite entre 100,100 et 200,200
```



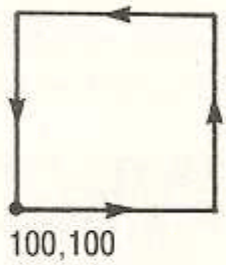
DRAWR DX,DY spécifie un déplacement relatif au point courant :

```
10 MODE 2
20 PLOT 100,100,1
30 DRAWR 100,100,1              ' droite entre 100,100 et 200,200
```

Les programmes ci-dessous représentent un carré de 100,100 en absolu et en relatif :

```
10 '------------carre
20 MODE 1
30 PLOT 100,100,1              ' Positionne en 100,100
40 DRAW 200,100,1              ' droite avec le Point 200,100
45 DRAW 200,200,1
50 DRAW 100,200,1
60 DRAW 100,100,1

5 '------------- carre en relatif
10 MODE 1
20 PLOT 100,100,1        ' Positionne en 100,100
30 DRAWR 100,0,1         ' droite dx=100 dy=0
40 DRAWR 0,100,1         ' droite dx=0 dy=100
50 DRAWR -100,0,1
60 DRAWR 0,-100,1
```



Le programme ci-dessous représente un rectangle plein :

```
10 '--------- rectangle
20 MODE 1
30 xa=100:ya=100       ' origine
40 l=200                ' longueur
50 h=100               ' hauteur
60 FOR x=xa TO xa+l
70  PLOT x,ya,1
80  DRAW x,ya+h,1
90 NEXT x
```



Remarque importante :

Lorsque le numéro de "stylo" n'est pas spécifié dans l'instruction PLOT ou DRAW, le stylo par défaut est le dernier utilisé. Par conséquent, si un programme précédent a utilisé le stylo 0, le programme en cours ne trace rien. Il est donc prudent de spécifier le numéro de stylo. Il est également recommandé d'initialiser la table des couleurs qui pourrait être modifiée par un programme précédent.

```
10 INK 0,1 : INK 1,24
```

MOVE X, Y, stylo
MOVER dx, dy, stylo

Positionne le curseur graphique (comme PLOT et PLOTR) mais sans affichage.

XPOS/YPOS

Fournit la position du curseur graphique.

```
10 MODE 1
20 MOVE 100,100
30 DRAW 150,140
50 PRINT XPOS,YPOS

150    140
```

CLG papier

Efface l'écran graphique. Positionne le curseur en bas de l'écran graphique.

CLG 2

TEST (X,Y)
TESTR (X,Y)

Donne le numéro de stylo qui a été utilisé pour le point X,Y.

```
10 MODE 1:PEN 1:PAPER 0
20 PLOT 100,100,1
30 PRINT TEST(100,100)

RUN

1
```

ORIGIN X,Y, gauche, droite, bas, haut

Redéfinit l'origine du curseur graphique. Les coordonnées spécifiées dans PLOT, DRAW et MOVE sont relatives à la nouvelle origine.

Ci-dessous l'origine est redéfinie en 100,100. DRAW 30,30 trace une droite entre la nouvelle origine et le point 130,130.

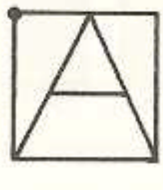
```
10 MODE 1
20 ORIGIN 100,100      ' redefinit l'origine
30 DRAW 30,30          ' trace une droite a partir de 100,100
```

Si les paramètres gauche, droite, bas, haut sont spécifiés, une fenêtre graphique est définie.

```
10 '--- FENETRE GRAPHIQUE
20 MODE 1
30 INK 1,0:INK 0,26
40 '-- Origine 440,100
50 ORIGIN 440,100,440,640,100,300
60 CLG 2
70 '--
80 DRAW 200,200,3
90 MOVER -5,-2
100 FILL 3               CPC664 SEULEMENT
```

TAG n° fenêtre
TAGOFF n° fenêtre

Pour afficher du texte à la position du curseur graphique, on utilise TAG. Le texte est affiché sous le curseur graphique.



Si l'instruction PRINT n'est pas suivie d'un point-virgule, les caractères CHR$(10) et CHR$(13) sont affichés à la suite du texte.

```
5 '-------- affichage avec TAG
10 MODE 1
20 TAG
30 FOR x=100 TO 200 STEP 10
40  MOVE x,x
50  PRINT "AMSTRAD";
60 NEXT x
```

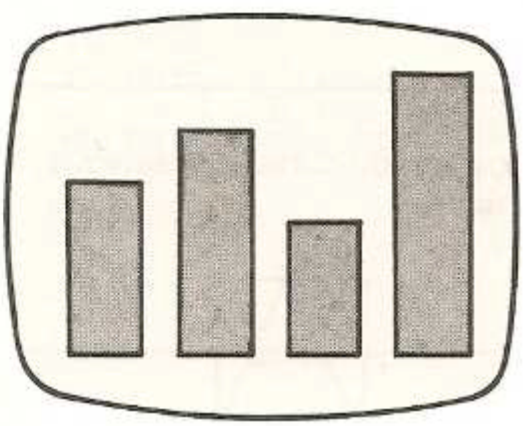


TAGOFF annule l'effet de TAG ; l'affichage par PRINT se fait normalement à partir du curseur texte.

Voici divers exemples de programmes utilisant la haute résolution.

Ce programme affiche un histogramme de 4 valeurs définies en DATA.

```
10 '------------- histogramme
20 MODE 1
30 x0=20:y0=100                ' origine
40 lg=15                       ' largeur
50 itv=20                      ' intervalle
60 DATA 50,70,30,80
70 FOR i=1 TO 4
80  READ h
90   xb=x0+(i-1)*itv
100  ce=INT(RND(1)*3)+1        ' couleur
110  FOR dx=1 TO lg            ' rectangle
120   PLOT xb+dx,y0,ce:DRAW xb+dx,y0+h,ce
130  NEXT dx
140 NEXT i
150 '-------- axes
160 PLOT x0,y0,1:DRAW x0+100,y0
170 PLOT x0,y0,1:DRAW x0,y0+100
```



□ Le programme ci-dessous affiche les 15 premières couleurs de la table des couleurs en MODE 0.

```
10 '------------ couleurs
20 ' trace 15 boites de couleurs differentes
30 ' essayer avec mode 1
40 '
50 MODE 0
60 INK 0,1:INK 1,24:PAPER 0:PEN 1
70 x0=20:y0=100                  ' origine
80 lg=15                         ' largeur
90 itv=20                        ' intervalle
100 FOR i=1 TO 15
110   xb=x0+(i-1)*itv
120  ce=i                        ' couleur
130  FOR dx=1 TO lg              ' rectangle
140   PLOT xb+dx,y0,ce:DRAW xb+dx,y0+100,ce
150  NEXT dx
160 NEXT i
```



□ Anneau

En traçant un cercle plein en couleur de fond sur un cercle plein, nous représentons un anneau.

```
10 '------------------- anneau
20 MODE 1
30 INK 0,1:INK 1,24:PAPER 0:PEN 1
40 x0=100:y0=100:r=60:ce=1:GOSUB 80   ' 1er cercle
50 r=40:ce=0:GOSUB 80                 ' 2eme cercle
60 END
70 '------------------- cercle
80 r2=r*r
90 FOR dx=-r TO r
100  dy=SQR(r2-(dx*dx))
110  PLOT x0+dx,y0+dy:DRAW x0+dx,y0-dy,ce
120  NEXT dx
130 RETURN
```



□ Croissant

Avec le même principe que ci-dessus, nous représentons un croissant.

```
10 '-------------- croissant
20 MODE 1
25 INK 0,1:INK 1,24:PAPER 0:PEN 1
30 x0=100:y0=100:r=60:ce=1:GOSUB 70   ' 1er cercle
40 x0=160:r=80:ce=0:GOSUB 70          ' 2eme cercle
50 END
60 '------------------- cercle
70 r2=r*r
80 FOR dx=-r TO r
90  dy=SQR(r2-(dx*dx))
100  PLOT x0+dx,y0+dy:DRAW x0+dx,y0-dy,ce
110  NEXT dx
120 RETURN
```



□ Ours

Le tracé de l'ellipse est réalisé en faisant pivoter une droite autour du centre. Le "pas" dépend de la grandeur de l'ellipse.

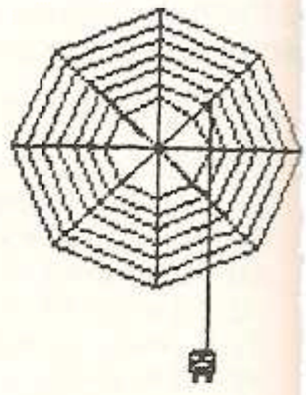
```
10 '------------------ ours
20 MODE 1
30 INK 0,1:INK 1,24:PAPER 0:PEN 1
40 x0=100:y0=100
50 x1=x0:y1=y0:r=60:ce=1:GOSUB 180          ' tete
60 X1=X0-40:Y1=Y0+50:R=40:GOSUB 180         ' oreilles
70 X1=X0+40:Y1=Y0+50:R=40:GOSUB 180
80 x1=x0:y1=y0-10:r=10:ce=0:GOSUB 180       ' nez
90 x1=x0-20:y1=y0+20:r=6:GOSUB 180          ' yeux
100 x1=x0+20:y1=y0+20:r=6:GOSUB 180
110 PLOT x0-4,y0-32:DRAW x0+2,y0-32,0
120 '
130 FOR y=-14 TO 14
140  PLOT x0-30,y0-70+y:DRAW x0+30,y0-70-y,1
150 NEXT y
160 END
170 '------------------------------ ellipse
180 h=r*2:l=r*2.3
190 FOR a=0 TO 2*PI+0.1 STEP 1.2/r
200  x=x1+(l/2)*COS(a)
210  y=y1+(h/2)*SIN(a)
220  PLOT x1,y1:DRAW x,y,ce
230 NEXT a
240 RETURN
```



□ Palmier

```
10 '--------------------- Palmier
20 MODE 2
30 DEG
40 ORIGIN 300,20
50 MOVE 9,8
60 FOR a=1  TO 20
70   FOR b= -90 TO 90 STEP 30
80   DRAW SIN(b)*LOG(20-a+1)*8-10,COS(b)*8*LOG(20-a+1)+a*11
90   NEXT b
100 NEXT a
105 '------------------
110 RAD
120 a$="Pascale"
130 b=LEN(a$)
135 TAG
140 FOR c=1 TO b
150 FOR d=90 TO 270 STEP 14
160     x=SIN(d)*(c*3)*8-10
170     y=COS(d)*(c*2)*8+28*8
180     MOVE x,y:PRINT MID$(a$,c,1);
190  NEXT d
200 NEXT c
```

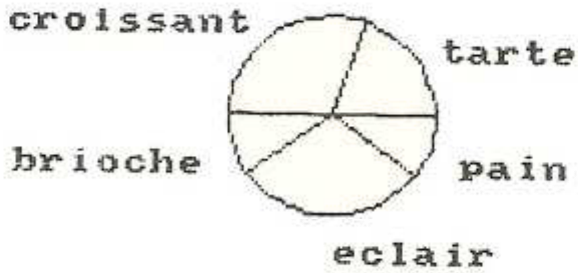
□ Araignée



```
20 MODE 1:INK 0,1:INK 1,24
30 X0=300 :Y0 =200
40 R=30
50 '
60 FOR M=1 TO 3 STEP 0.3
70  PLOT x0+r*m,y0,1
80  FOR A=PI/4 TO 2*PI+0.1 STEP PI/4
90   X=R*M*COS(A)+X0
100   Y=R*M*SIN(A)+Y0
110   DRAW X,Y,1
120   PLOT X0,Y0: DRAW X,Y
130  NEXT A
140 NEXT M
150 '
160 PLOT X0+R,Y0+80,1:DRAW X0+R,Y0+R
170 LOCATE 21,21:PRINT CHR$(225)
180 LOCATE 21,22:PRINT CHR$(162)
190 LOCATE 20,22:PRINT CHR$(192);CHR$(196);CHR$(193);
200 FOR a=1 TO 100:NEXT
210 LOCATE 20,22:PRINT CHR$(193);CHR$(145);CHR$(192);
220 FOR a=1 TO 100:NEXT
230 GOTO 190
```

□ Histogramme circulaire

```
20 MODE 2
30 INK 0,1:INK 1,24:PAPER 0:PEN 1
40 h(1)=0.2:h$(1)="tarte"
50 h(2)=0.3:h$(2)="croissant"
60 h(3)=0.1:h$(3)="brioche"
70 h(4)=0.3:h$(4)="eclair"
80 h(5)=0.1:h$(5)="pain"
90 '
100 xa=200:ya=200
110 r=60
120 '
130 PLOT xa+r,ya
140 FOR a=0 TO 2*PI+0.1 STEP 0.1
150  x=xa+r*COS(a)
160  y=ya+r*SIN(a)
170  DRAW x,y,1
180 NEXT a
190 '------------------- affichage texte
200 aa=0
210 FOR p=1 TO 5
220  a=aa+PI*2*h(p)
230  x=xa+r*COS(a):y=ya+r*SIN(a)
240  PLOT xa,ya:DRAW x,y,1
250 '
260  at=aa+PI*h(p)          ' affichage texte
270  x=xa+r*1.3*COS(at):y=ya+r*1.3*SIN(at)
280  IF at>PI/2 AND at<3*PI/2 THEN x=x-8*LEN(h$(p))
290  PLOT x,y:TAG:PRINT h$(p);
300  aa=a
310 NEXT p
```

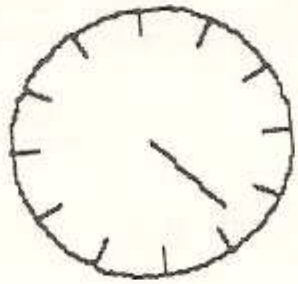


□ Chronomètre

```
20 MODE 2
30 INK 0,1:INK 1,24:PAPER 0:PEN 1
40 X0=300 :Y0=200:R=100              ' centre
50 ORIGIN x0,y0                      ' redefinition origine
60 GOSUB 240
70 '------------------ graduations
80 FOR A=0 TO 2*PI STEP PI/6
90  X=R*0.8*COS(A):Y=R*0.8*SIN(A)
100  X1=R*COS(A):Y1=R*SIN(A)
110  PLOT X,Y,1:DRAW X1,Y1,1
120 NEXT A
130 '----------------------- mouvement aiguille
140 R=R*0.7
150 FOR A=100 TO 0 STEP -PI/30
160  x=r*COS(a):y=r*SIN(a)
170  PLOT 0,0:DRAW x,y,1
180  SOUND 1,500,2
190  FOR TP=1 TO 120:NEXT TP
200  DRAW 0,0,0
210 NEXT A
220 END
```

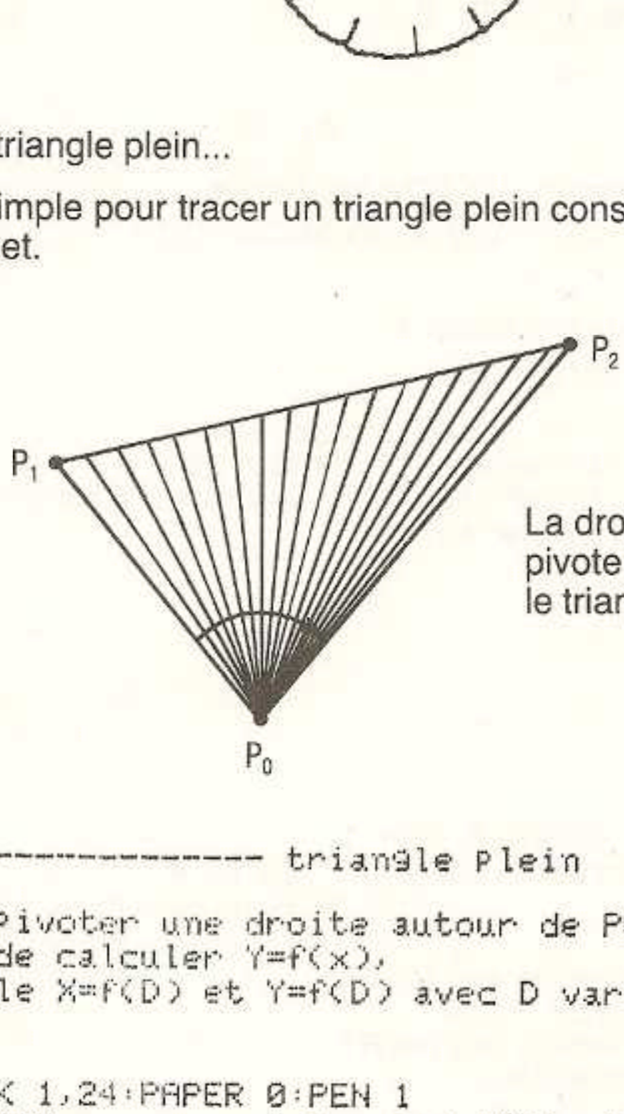
→

```
230 '------------------------- cercle
240 PLOT r,0
250 FOR A=0 TO 2*PI +0.2 STEP 0.05
260  DRAW r*COS(a),r*SIN(a),1
270 NEXT A
280 RETURN
```



□ Pour tracer un triangle plein...

La façon la plus simple pour tracer un triangle plein consiste à faire pivoter une droite autour d'un sommet.

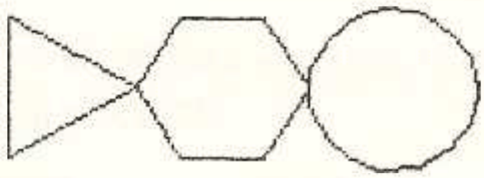


La droite P_0-P_1 pivote pour remplir le triangle.

```
10 '---------------------- triangle plein
15 '
20 ' On fait pivoter une droite autour de P0 entre P1 et P2.
30 ' Au lieu de calculer Y=f(x),
40 ' on calcule X=f(D) et Y=f(D) avec D variant avec un pas de 1
50 '
60 MODE 2
70 INK 0,1:INK 1,24:PAPER 0:PEN 1
80 x0=140:y0=100              ' 1er point(pivot)
90 x1=210:y1=230              ' 2eme point
100 x2=120:y2=200             ' 3eme point
110 '
120 d=SQR((y2-y1)^2+(x2-x1)^2)' distance entre P1 et P2
130 IF d=0 THEN END
140 cx=(x2-x1)/d:cy=(y2-y1)/d
150 FOR dd=0 TO d             ' on parcourt la droite entre P1
160  x3=x1+dd*cx:y3=y1+dd*cy    et P2
170  PLOT x0,y0,ce:DRAW x3,y3,1
180 NEXT dd
190 RETURN
```

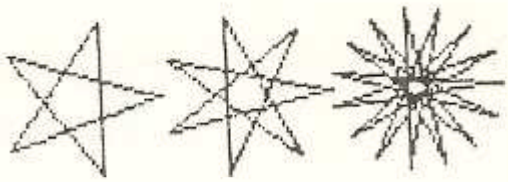
□ Figures inscrites dans un cercle

```
10 '------------------------ figures inscrites dans un cercle
20 '
30 MODE 2:INK 0,1:INK 1,24:PAPER 0:PEN 1
40 xc=100:yc=100                        ' centre
50 r=50                                 ' rayon
60 ncot=3                               ' nombre de cotes
70 GOSUB 140
80 '-------------------
90 ncot=6:xc=200:GOSUB 140             ' hexagone
100 '-------------------
110 ncot=15:xc=300:GOSUB 140           ' cercle
120 END
130 '--------------------------- sp9m
140 PLOT xc+r,yc
150 FOR c=1 TO ncot
160  a=c*PI*2/ncot
170  x=xc+r*COS(a)
180  y=yc+r*SIN(a)
190  DRAW x,y,1
200 NEXT c
210 RETURN
```



□ Etoiles

```
10 '------------------------ trace d'etoiles
20 '
30 MODE 2:INK 0,1:INK 1,24:PAPER 0:PEN 1
40 xc=100:yc=100                        ' centre
50 r=50                                 ' rayon
60 ncot=5                               ' nombre de branches
70 GOSUB 140
80 '-------------------
90 ncot=7:xc=200:GOSUB 140
100 '-------------------
110 ncot=15:xc=300:GOSUB 140
120 END
130 '--------------------------- sp9m
140 PLOT xc+r,yc
150 FOR c=1 TO ncot
160  a=c*(ncot-1)*PI/ncot
170  x=xc+r*COS(a)
180  y=yc+r*SIN(a)
190  DRAW x,y
200 NEXT c
210 RETURN
```



200 NEXT c
210 RETURN

□ Recopie d'écran

Nous proposons ci-dessous un programme de recopie d'écran haute résolution pour l'imprimante AMSTRAD.

```
2000 '---------------- recopie d'ecran sur imprimante AMSTRAD
2010 DEFINT a-z
2020 DIM lg(320)
2030 ln=399
2040 FOR lg=1 TO 30
2050  FOR x=0 TO 319
2060   lg(x)=0
2070   FOR p=0 TO 6
2080     IF TEST(x*2,ln-p*2)>0 THEN lg(x)=lg(x)+(2^p)
2090   NEXT p
2100  NEXT x
2110  PRINT #8,CHR$(27);CHR$(75);CHR$(2);CHR$(64);
2120  FOR z=0 TO 319:PRINT #8,CHR$(lg(z));:NEXT z
2130  PRINT #8,CHR$(lg(320))
2140  ln=ln-14
2150 NEXT lg
2160 PRINT #8,CHR$(15)
```

Pour obtenir une copie d'écran, faire :

MERGE "HARDC" puis "RUN 2000"

Il est prévu pour fonctionner avec PAPER 0. En revanche, la couleur d'écriture définie dans PEN n'a pas d'importance.

LA REDÉFINITION DES CARACTÈRES

- **SYMBOL**
- **SYMBOL AFTER**

SYMBOL

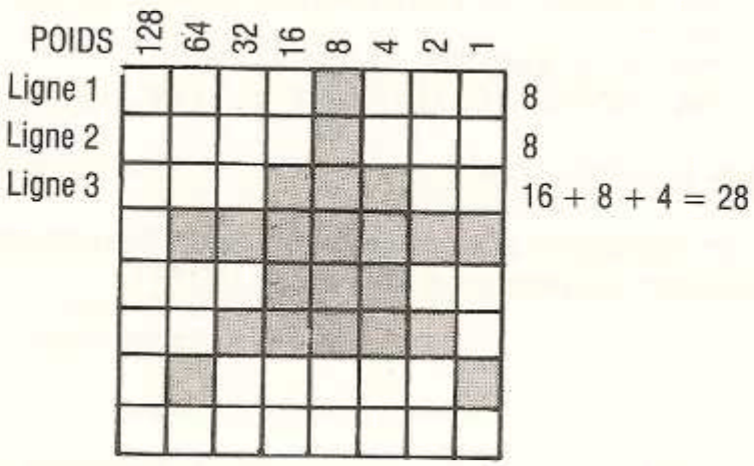
Les caractères standard peuvent être modifiés à l'aide de l'instruction **SYMBOL**. Ceci permet notamment de représenter des formes qui sont ensuite affichées et effacées très rapidement.

La règle d'utilisation est simple :

- On dessine le caractère dans une matrice de 8 × 8 points.
- Pour chaque ligne, on effectue la somme des "poids" des points de la ligne qui doivent être allumés.

Exemple :

Sur l'exemple ci-dessous, pour la ligne 3, on obtient :
16 + 8 + 4 = 28.



Dans une instruction **SYMBOL**, on indique la suite des nombres obtenus dans l'ordre des lignes.

```
10 '---------- etoile
20  MODE 1
30 c=150          ' caractere a modifier
40 SYMBOL AFTER c
50 SYMBOL c,8,8,28,127,28,34,65,0
60 PRINT CHR$(c)
```

SYMBOL AFTER

"SYMBOL AFTER" spécifie le premier code des caractères à modifier.

Ci-dessous une étoile traverse l'écran de gauche à droite, puis de droite à gauche. La vitesse s'adapte grâce à la temporisation en ligne 170. L'effacement se fait en imprimant un espace devant le caractère.

```
10 '---------→ deplacement etoile
20 MODE 1
30 INK 0,26              ' fond blanc
40 INK 1,0               ' encre noire
50 PAPER 0:PEN 1
60 c=145                 ' caractere redefini
70 SYMBOL AFTER c
80 SYMBOL c,8,8,28,127,28,34,65,0
90 '------------- deplacement etoile
100 y=10:x1=1:x2=36          ' bornes
110 s=1                      ' sens
120 '
130 FOR x=x1 TO x2 STEP s
140  LOCATE x,y:PRINT CHR$(32);CHR$(c);CHR$(32);
150  FOR tp=1 TO 80:NEXT tp
160 NEXT x
170 x=x1:x1=x2:x2=x:s=-s    ' inversion
180 GOTO 130
```

La forme ci-dessous est représentée avec deux caractères redéfinis.



```
10 '------------ chien
20 MODE 1
30 c=145
40 SYMBOL AFTER c
50 SYMBOL c,68,56,56,7,7,7,4,12
60 SYMBOL c+1,0,4,8,240,240,240,16,48
70 '
75 chien$=CHR$(c)+CHR$(c+1)
80 LOCATE 10,10:PRINT chien$
```

Utilisation de TAG :

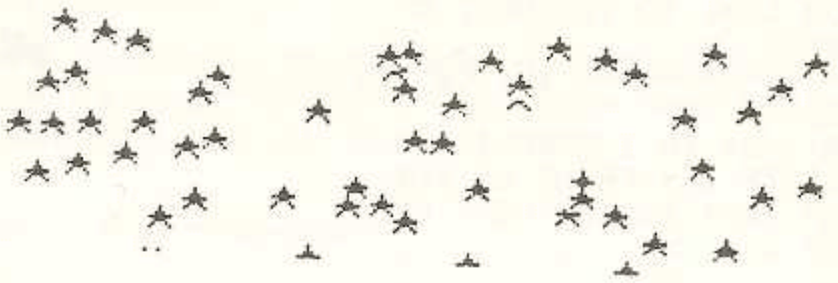
Pour positionner un caractère avec le curseur graphique, on utilise TAG. Naturellement, le positionnement est plus précis qu'avec LOCATE.

Exemples :

□ Ciel étoilé

```
10 '----------- ciel etoile
20 MODE 1
30 INK 0,26              ' fond blanc
40 INK 1,0               ' encre noire
50 PAPER 0:PEN 1
60 c=125                 ' caractere redefini
70 SYMBOL AFTER c
80 SYMBOL c,8,8,28,127,28,34,65,0
90 TAG
100 FOR e=1 TO 100
110  x=RND(1)*500
120  y=RND(1)*300
130  MOVE x,y
140  PRINT CHR$(c);
150 NEXT e
```

→



□ Etoile (avec TAG)

Ci-dessous, nous déplaçons une étoile. Le déplacement est plus progressif qu'avec LOCATE.

```
10 '------------------ deplacement etoile (avec TAG)
20 MODE 1
30 INK 0,26              ' fond blanc
40 INK 1,0               ' encre noire
50 PAPER 0:PEN 1
60 c=125                 ' caractere redefini
70 SYMBOL AFTER c
80 SYMBOL c,8,8,28,127,28,34,65,0
90 '------------- deplacement etoile
100 TAG
110 FOR x=1 TO 500 STEP 2
120  MOVE  x,200
130  PRINT CHR$(32);CHR$(c);
140 NEXT x
```

□ Formes de 16 × 16 points

Pour représenter des formes de 16 × 16 points, il faut redéfinir 4 caractères.

Ci-dessous, nous représentons une locomotive :

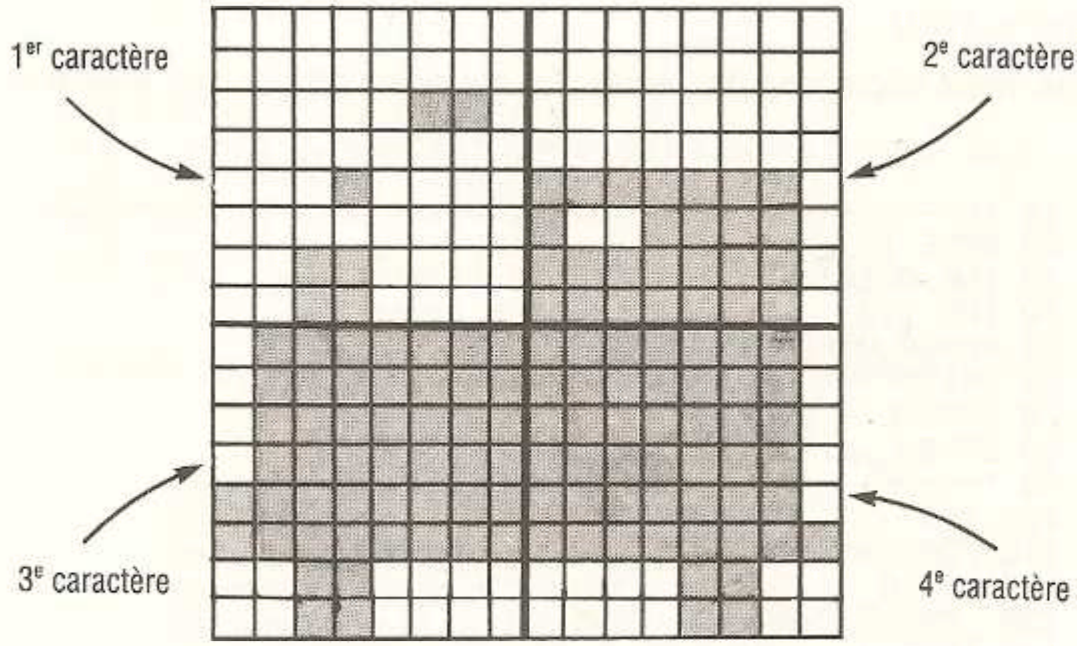
CHR$(10) provoque un saut de ligne,
CHR$(8) décale le curseur vers la gauche.

```
10 '------------------- locomotive
20 '
30 MODE 1:INK 0,1:INK 1,24
40 C=145                    ' 1er caractere a modifier
50 SYMBOL AFTER C
60 SYMBOL c,0,0,6,0,16,0,48,48
70 SYMBOL c+1,0,0,0,0,254,198,254,254
80 SYMBOL c+2,127,127,127,127,255,255,48,48
90 SYMBOL c+3,254,254,254,254,255,255,24,24
100 l$=CHR$(c)+CHR$(c+1)+CHR$(10)+CHR$(8)+CHR$(8)+CHR$(c+2)+
CHR$(c+3)+CHR$(11)
110 LOCATE 10,2:PRINT l$
120 t$=l$+l$                      ' train
130 LOCATE 10,10:PRINT t$
140 '--- chaine effacement
150 ef$=CHR$(32)+CHR$(10)+CHR$(8)+CHR$(32)
160 '----------- avance locomotive
170 y=18
180 FOR x=24 TO 1 STEP-1
190  LOCATE x,y:PRINT l$;ef$
```

→

```
200  FOR tp=1 TO 100:NEXT tp
210 NEXT x
220 '-------------- avance train
230 y=23
240 FOR x=24 TO 1 STEP-1
250  LOCATE x,y:PRINT t$;ef$
260  FOR tp=1 TO 100:NEXT tp
270 NEXT x
```



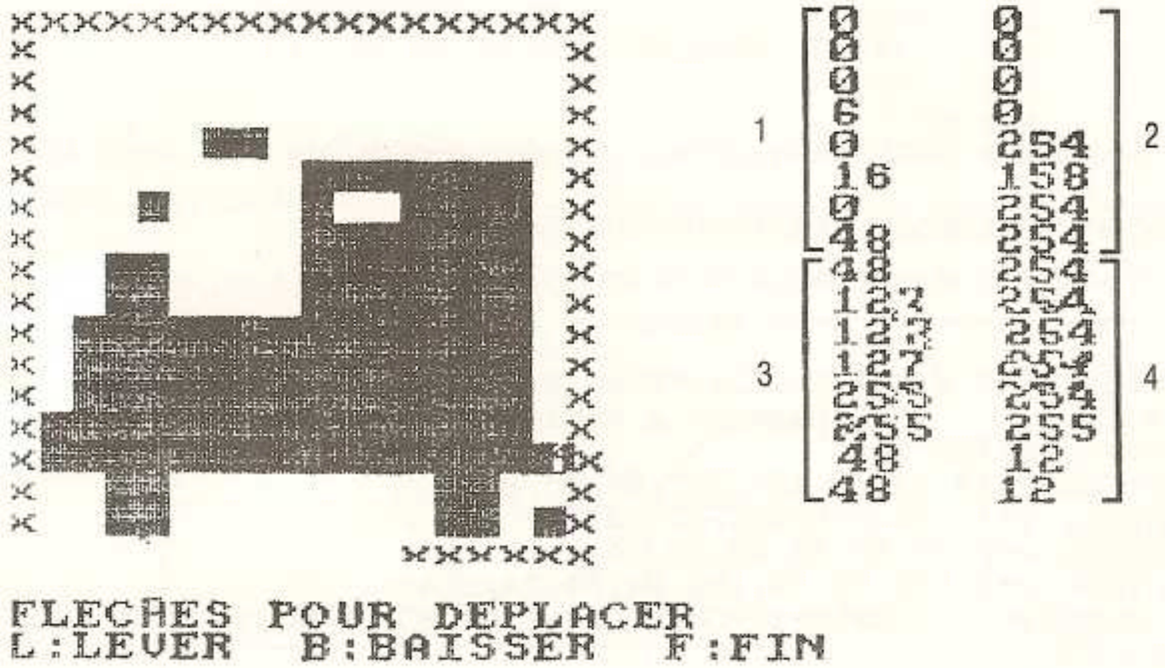


Pour déplacer la locomotive plus progressivement qu'avec LOCATE, nous utilisons **TAG**. La chaîne "h$" représente la partie supérieure de la locomotive et "b$" la partie inférieure.

```
10 '-------- deplacement locomotive avec TAG
20 '
30 MODE 1:INK 0,1:INK 1,24
40 C=145                       ' 1er caractere a modifier
50 SYMBOL AFTER C
60 SYMBOL c,0,0,6,0,16,0,48,48
70 SYMBOL c+1,0,0,0,0,254,198,254,254
80 SYMBOL c+2,127,127,127,127,255,255,48,48
90 SYMBOL c+3,254,254,254,254,255,255,24,24
100 '
110 h$=CHR$(c)+CHR$(c+1)+CHR$(32)          ' haut
120 b$=CHR$(c+2)+CHR$(c+3)+CHR$(32)        ' bas
130 y=100
140 TAG
150 FOR x=500 TO 1 STEP -2
160  MOVE x,y:PRINT h$;:MOVE x,y-14:PRINT b$;
180 NEXT x
```

□ Générateur de caractères

Nous proposons un "générateur de caractères" permettant de représenter des formes de 16 × 16 points. En mode "Télécran", nous représentons la forme. Le programme fournit ensuite les valeurs décimales des 4 caractères.




```
10 '--------- GENERATEUR DE CARACTERES 16*16
20 MODE 1
30 DIM t(18,18)
40 FOR x=1 TO 18:LOCATE x,1:PRINT "x":LOCATE x,18:PRINT "x":NE
XT x
50 FOR y=1 TO 18:LOCATE 1,y:PRINT "x":LOCATE 18,y:PRINT "x":NE
XT y
60 LOCATE 1,20:PRINT "FLECHES POUR DEPLACER"
70 LOCAIE 1,21:PRINT "L:LEVER  B:BAISSER  F:FIN"
80 x=10:y=10
90 '---------
100 c$=INKEY$:IF c$<>"" THEN 160
110 LOCATE x,y:PRINT CHR$(143)
120 FOR tp=1 TO 10:NEXT tp
130 LOCATE x,y:PRINT CHR$(32)
140 GOTO 100
150 '
160 IF cs=0 THEN LOCATE x,y:PRINT CHR$(143):t(x-1,y-1)=1
170 IF cs=1 THEN LOCATE x,y:PRINT CHR$(32):t(x-1,y-1)=0
180 '
190 c$=UPPER$(c$)
200 c=ASC(c$)
210 IF c=242 THEN IF x>2 THEN x=x-1
220 IF c=243 THEN IF x<17 THEN x=x+1
230 IF c=241 THEN IF Y<17 THEN y=y+1
240 IF c=240 THEN IF Y>2 THEN y=y-1
250 IF c$="L" THEN cs=1
260 IF c$="B" THEN cs=0
270 IF c$="F" THEN 300
280 GOTO 100
290 '------------------ calcul valeurs decimales
```

→

```
300 FOR CL=1 TO 2       ' 2 colonnes
310  FOR L=1 TO 16      ' 16 lignes
320   PS=(CL-1)*8
330   ND=0               ' valeur decimale
340   FOR X=1 TO 8      ' 1 caractere
350    A=0:IF T(X+PS,L)=1 THEN A=1
360    ND=ND+A*2^(8-X)
370   NEXT X
380   LOCATE 20+CL*5,L:PRINT ND;SPC(1)
390   NEXT L
400 NEXT CL
410 GOTO 100
```

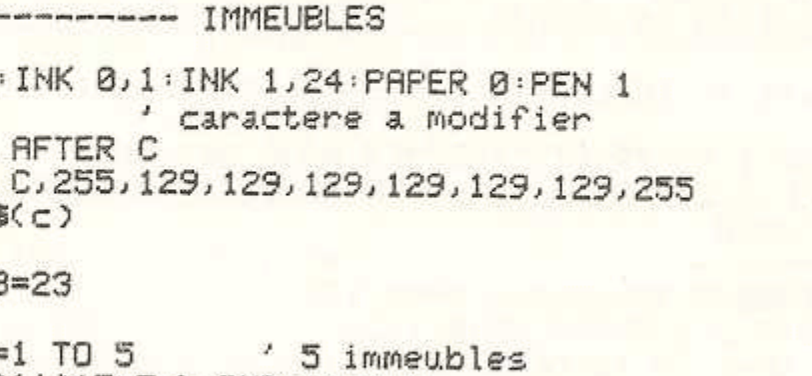
□ Le programme ci-dessous représente une maison.

```
10 '------------------- maison
20 '
30 MODE 1:INK 0,1:INK 1,24:PAPER 0:PEN 1
40 C=125         ' caractere a modifier
50 SYMBOL AFTER C
60 SYMBOL c,12,12,63,127,255,64,95,85
70 SYMBOL c+1,0,0,248,252,255,2,2,2
80 SYMBOL c+2,95,85,85,85,95,64,64,127
90 SYMBOL c+3,122,74,74,106,74,74,74,254
100 m$=CHR$(c)+CHR$(c+1)+CHR$(10)+CHR$(8)+CHR$(8)+CHR$(c+2)+
CHR$(c+3)
110 PRINT m$
```



□ Celui-ci des immeubles dont la hauteur est aléatoire.

```
10 '-------------- IMMEUBLES
20 '
30 MODE 1:INK 0,1:INK 1,24:PAPER 0:PEN 1
40 C=125         ' caractere a modifier
60 SYMBOL AFTER C
70 SYMBOL C,255,129,129,129,129,129,129,255
75 c$=CHR$(c)
80 '
90 XB=1:YB=23
100 '
110 FOR N=1 TO 5        ' 5 immeubles
130  H=RND(1)*5+5:L=RND(1)*3+3
140  FOR Y=YB TO YB-H STEP-1
150   FOR X=XB TO XB+L
160    LOCATE X,Y:PRINT C$
170   NEXT X
180  NEXT Y
190 XB=XB+L+2
200 NEXT N
```

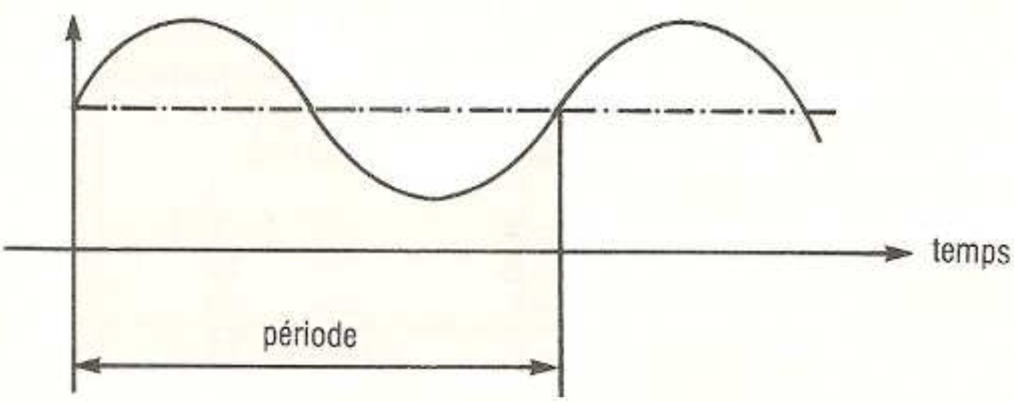


LES SONS

- SOUND
- ENV
- ENT
- SQ
- ON SQ GOSUB
- RELEASE

SOUND canal, période, durée, volume, enveloppe volume, enveloppe fréquence, période bruit

Un son est défini par sa période, son volume et sa durée. Plus la période est grande, plus le son est grave.



Exemple :

SOUND 1,478 ' canal,période

joue la note DO pendant 0,2 seconde. Pour jouer la note pendant 1 seconde, on écrit :

SOUND 1,478,100 ' temps=1 sec.

Le temps est spécifié en 1/100 de seconde. Le volume spécifié doit être compris entre 0 et 7 si aucune enveloppe n'est spécifiée. Il est de 4 par défaut.

SOUND 1,478,50,6 ' temps=.5 sec./volume=6

La fréquence en Hertz s'obtient par :

fréquence=125000/période spécifiée

Pour obtenir un son de fréquence F, on calcule la période à spécifier par : 125000/F

Exemple :

Pour obtenir du 1000 Hertz, la période spécifiée doit être 125.

```
10 INPUT "Frequence ";f
20 p=125000/f
30 SOUND 1,p,100
40 goto 10
```

L'instruction SOUND ne bloque pas l'exécution du programme. Plusieurs sons peuvent être joués simultanément sur trois canaux (A, B, C).

```
10 SOUND 1,125,100        ' canal A  1000 HERTZ pendant 1 seconde
20 SOUND 2,500,200        ' canal B  250 HERTZ pendant 2 secondes
30 PRINT  "SUITE"
```

Avec une seule commande SOUND, on peut jouer la même note sur plusieurs canaux grâce à un système de codage.

A → 1
B → 2
C → 4

Exemple :

Pour jouer sur A et C, on code 5 (1+4) dans l'instruction SOUND. Chaque canal dispose d'une file d'attente pouvant comporter jusqu'à quatre commandes SOUND.

La synchronisation entre des canaux se spécifie en codant :

Synchronisation avec A	→	8
Synchronisation avec B	→	16
Synchronisation avec C	→	32
Attente	→	64
Libère	→	128

Période :

La période spécifiée dans SOUND doit être comprise entre 0 et 4095. 0 spécifie l'absence de fréquence.

Durée :

La durée qui est par défaut 20 (0.2 sec.) doit être comprise entre −32768 et +32767. Si la durée spécifiée est nulle, la durée est fournie par la commande **ENV**. Une valeur négative donne le nombre de répétitions de l'enveloppe de volume.

Volume :

Il doit être compris entre 0 et 15. **Sans enveloppe de volume**, il est égal à 4 par défaut et doit être compris entre 0 et 7. **Avec enveloppe de volume**, il est égal à 12 par défaut.

Enveloppe de volume :

Un son peut être "modulé" en amplitude par une enveloppe (de période plus importante). Cf. instruction ENV.

Enveloppe de ton :

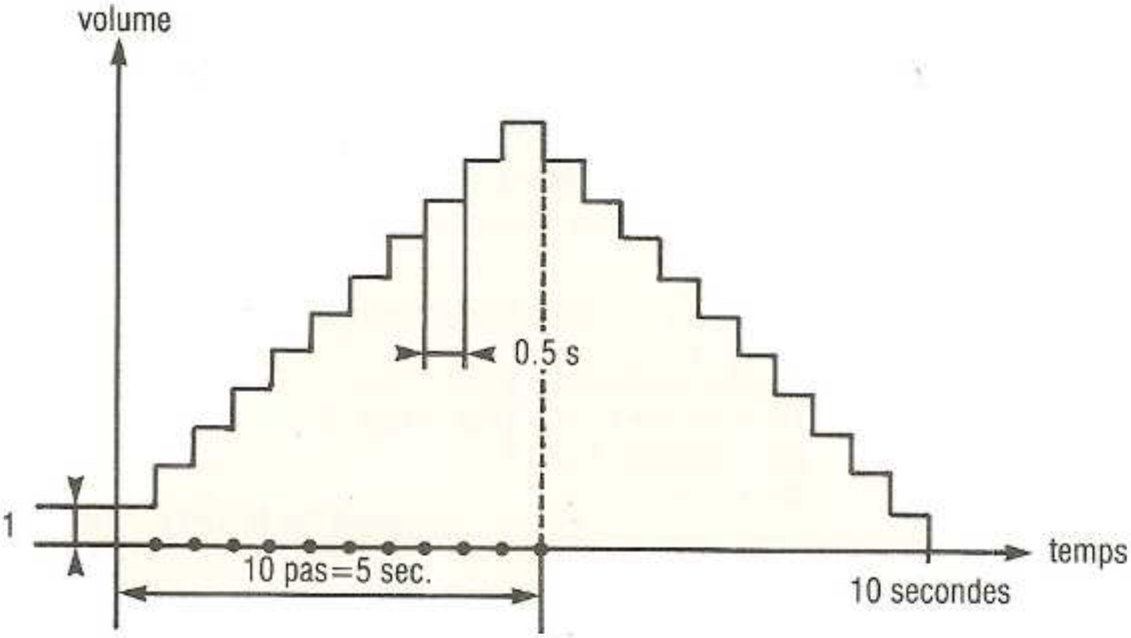
Cf. instruction ENT.

ENV numéro, nombre pas, pas volume, temps, nombre pas, pas volume, temps...

Pour définir une enveloppe de volume, on indique le nombre de pas, le pas et le temps de pause (en 1/100 sec.).

Exemple :

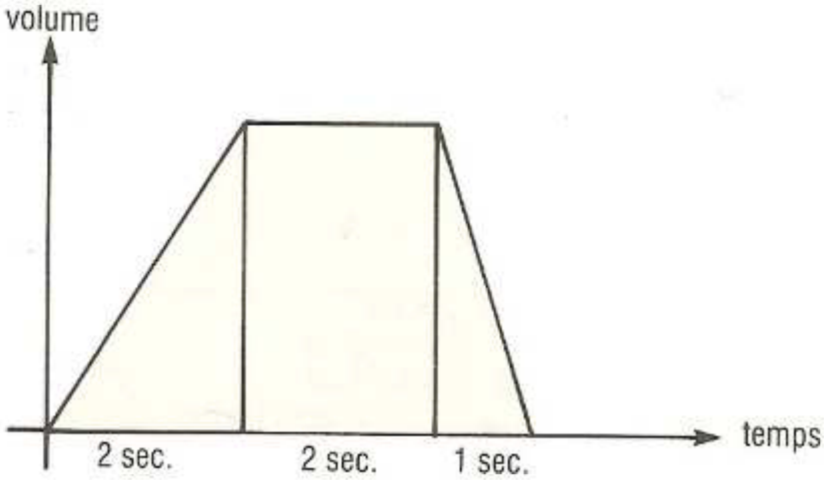
Nous définissons une enveloppe croissante pendant 5 secondes puis décroissante pendant 5 secondes.




```
2000 ENV 1,10,1,50,10,-1,50      ' 10 Pas de 1 de .5 sec
2010 SOUND 1,125,1000,1,1        ' Pendant 10 secondes
```

Pour obtenir l'enveloppe ci-dessous, nous écrivons :

```
2000 ENV 1, 5,2,20, 1,0,200, 10,-1,20
2010 SOUND 1,125,500,1,1
```



ENT numéro, nombre pas, pas période, temps, nombre pas, pas période, temps...

Cette instruction permet de faire varier la fréquence de l'instruction SOUND pendant son exécution.

Dans l'exemple ci-dessous, la période de 125 au départ devient : 125+100*2

```
2000 ENT 1,100,2,3                  ' 100 Pas de 2 Pendant 3/100 sec
2010 SOUND 1,125,300,15,0,1
```

Dans l'exemple ci-dessous, la période augmente pendant 50*3 centièmes de seconde et diminue pendant 50*3 centième de secondes.

```
2000 ENT 1, 50,2,3, 50,-2,3
2010 SOUND 1,25,300,15,0,1      ' 3 sec
```

Exemples divers :

□ Génère des sons aléatoires.

```
10 SOUND 1,RND(1)*400,3,15
20 GOTO 10
```

□ Tir de laser.

```
5 '--------- laser
10 FOR p=1 TO 100 STEP 5
20  SOUND 1,p,1
30 NEXT p
```

□ Sirène de pompier.

```
10 '--------- Pompier
20 SOUND 1,200,50
30 SOUND 1,100,50
40 GOTO 20
```

□ En appuyant simultanément sur A, S et D, la note "DO" est jouée avec 3 octaves différentes.

```
10 '- joue sur 3 canaux une note sur 3 octaves  avec touches A,S,D
20 '
30 ' appuyer sur A seul,S seul,D seul
40 ' Puis A et S simultanement
50 '
60 nt=478             ' note
70 d=5                ' duree
80 '
90 IF INKEY(69)=0 THEN SOUND 1,nt,d
100 IF INKEY(60)=0 THEN SOUND 2,nt/2,d
110 IF INKEY(61)=0 THEN SOUND 4,nt/4,d
120 GOTO 90
```

SQ (canal)

Donne l'état d'un canal.

Bits 0,1,2 : nombre de places libres
- 3 : synchronisation A
- 4 : synchronisation B
- 5 : synchronisation C
- 6 : attente
- 7 : canal occupé

□ Ci-dessous, 132 représente la somme de 128 (canal occupé) et de 4 (nombre de places libres).

```
10 SOUND 1,125,100                          ' 1 seconde
20 PRINT SQ(1);
30 GOTO 20

132  132  132  132 ...... 4 4 4 4
```

□ Le programme suivant joue sur le canal 1 pendant une seconde puis sur le canal 2.

```
5 '------------- frequence 1000 sur 1 puis frequence 500  sur 2
10 SOUND 1,125,100                         ' 1 seconde
20 IF (SQ(1) AND 128)<>128 THEN SOUND 2,250,100:END
30 GOTO 20
```

ON SQ (canal) GOSUB n° ligne

Provoque un branchement au numéro de ligne spécifié lorsqu'il y a une place libre dans le canal spécifié.

```
10 SOUND 1,125,200
20 SOUND 1,250,100
21 SOUND 1,500,100
22 SOUND 1,1000,100
23 SOUND 1,2000,100
24 SOUND 1,4000,100
30 ON SQ(1) GOSUB 50
40 GOTO 40
45 '--------------
50 PRINT "Il y a une place libre pour le canal 1"
60 RETURN
```

RELEASE n° canal

Libère un canal en attente.

RELEASE 2 ⟶ 2
RELEASE 3 ⟶ 1+2

Exemples divers :

□ Ci-dessous, des notes sont choisies au hasard.

```
10 '---------------- MUSIQUE ALEATOIRE
20 DATA 4186,do
30 DATA 4698,re
40 DATA 5274,mi
50 DATA 5587,fa
60 DATA 6271,sol
70 DATA 7040,la
80 DATA 7902,si
90 '
100 FOR n=1 TO 7
110  READ f(n),x$
120 NEXT n
130 '--------------
140 nt=INT(RND(1)*7)+1          ' note
150 dur=INT (RND(1)*60)+10      ' duree
160 oct=INT(RND(1)*3)+1         ' octave
170 '
180 SOUND 1,125000/f(nt)*4/oct,dur
190 GOTO 140
```

Les fréquences données dans le programme sont celles du niveau d'octave 4. On trouvera en annexe un tableau complet des fréquences pour les niveaux d'octaves allant de −1 à 4.

□ Le programme ci-dessous joue une note frappée au clavier. A chaque touche correspond une note avec un niveau d'octave. La note jouée est de durée fixe. Nous avons augmenté le délai de répétition pour les touches afin d'éviter qu'une note ne soit jouée deux fois de suite malencontreusement.

```
10 '------------ JOUE 1 NOTE A LA FOIS DE DUREE FIXE
20 MODE 2:DEFINT a-z
30 SPEED KEY 30,20
40 LOCATE 1,20:PRINT "q  w  e  r  t  y  u OCTAVE 4"
50 LOCATE 1,21:PRINT "a  s  d  f  g  h  j OCTAVE 3"
60 '----------------- octave 4
70 DIM f(21),cl$(21)
80 DATA 4186,do,Q,  4698,re,W,  5274,mi,E
90 DATA 5587,fa,R, 6271,sol,T, 7040,la,Y
100 DATA 7902,si,U
110 '----------------- octave 3
120 DATA 2093,do,A, 2349,re,S, 2637,mi,D
130 DATA 2793,fa,F, 3135,sol,G, 3520,la,H
140 DATA 3951,si,J
150 '
160 FOR n=1 TO 14:READ f(n),x$,cl$(n):f(n)=125000/f(n):NEXT n
170 '--------------
180 c$=INKEY$:IF c$="" THEN 180
190 c$=UPPER$(c$)
200 FOR i=1 TO 14
210  IF c$=cl$(I) THEN  SOUND 1,f(i),15
220 NEXT i
230 GOTO 180
```

Le temps de recherche de la note correspondant à la touche frappée peut être supprimé en utilisant les valeurs ASCII des touches pour déterminer l'adresse de rangement des notes dans la table f().

```
160 FOR n=1 TO 14:READ f,x$,c$:f(ASC(c$))=125000/f:NEXT n
170 '---------------
180 c$=INKEY$:IF c$="" THEN 180
190 c$=UPPER$(C$)
210 c=ASC(c$): SOUND 1,f(c),15
230 GOTO 180
```

□ Le programme ci-dessous permet de jouer trois notes de durée variable simultanément. Lorsque l'opérateur appuie sur une touche de façon permanente, toutes les commandes SOUND pour une touche pendant la durée d'une note doivent être envoyées vers le même canal. Un simple test de la disponibilité des canaux avec SQ n'est donc pas suffisant ; une même note serait envoyée sur plusieurs canaux.

```
10 '-------- JOUE 3 NOTES SIMULTANEMENT sur canal 1 et 2 et 4
20 MODE 2:DEFINT a-z
30 LOCATE 1,20:PRINT "q  w  e  r  t  y  u octave 4"
40 LOCATE 1,21:PRINT "a  s  d  f  g  h  j octave 3"
50 '------------------ octave 4
60 DIM f(21),cl(21),c(21)
70 DATA 4186,do,67,  4698,re,59,  5274,mi,58
80 DATA 5587,fa,50, 6271,sol,51, 7040,la,43
90 DATA 7902,si,42
100 '------------------- octave 3
110 DATA 2093,do,69, 2349,re,60, 2637,mi,61
120 DATA 2793,fa,53, 3135,sol,52, 3520,la,44
130 DATA 3951,si,45
140 '
150 FOR n=1 TO 14:READ f(n),x$,cl(n):f(n)=125000/f(n):NEXT n
160 '--------------
170 FOR i=1 TO 14
180  IF INKEY(cl(i))=0 THEN  230
190  IF c(i)<>0 THEN clib(c(i))=0:c(i)=0
200 NEXT i
210 GOTO 170
220 '
230 IF c(i)<>0 THEN c=c(i):SOUND c,f(i),15:GOTO 200
240 IF clib(1)=0 THEN c(i)=1:clib(1)=1:SOUND 1,f(i),15:GOTO 200
250 IF clib(2)=0 THEN c(i)=2:clib(2)=1:SOUND 2,f(i),15:GOTO 200
255 IF clib(3)=0 THEN c(i)=4:clib(3)=1:SOUND 4,f(i),15:GOTO 200
260 GOTO 200
```

programme

L'HORLOGE | 11

- TIME
- AFTER
- EVERY
- REMAIN
- DI
- EI

TIME

Fournit le temps écoulé depuis la mise sous tension en 1/300 seconde.

L'exemple ci-dessous affiche l'heure.

```
10 '-------- horloge
20 CLS
30 h=TIME
40 LOCATE 1,22
50 PRINT (TIME-h)/300
60 GOTO 40
```

L'exécution différée ou périodique de sous-programmes se programme avec les instructions AFTER (APRES) et EVERY (CHAQUE). C'est BASIC qui gère les chronomètres associés aux sous-programmes.

AFTER temps, n° chrono GOSUB n° ligne

Provoque l'exécution d'un sous-programme spécifié après le temps spécifié en 1/50e de seconde. Ci-dessous le message "COUCOU" est affiché après 2 secondes.

Le numéro de chrono spécifié doit être compris entre 0 et 3. En cas de conflit d'exécution, c'est le chrono 3 le plus prioritaire.

```
10 '-------- AFTER
20 MODE 1
30 AFTER 100,0 GOSUB 90                  ' apres 2 secondes
40 '
50 LOCATE 1,2:PRINT i:i=i+1:GOTO 50 ' boucle d'attente
60 END
80 '----------------------
90 LOCATE 1,22:PRINT "coucou"
100 RETURN
```

EVERY temps, n° chrono, GOSUB n° ligne

Provoque l'exécution périodique du sous-programme spécifié. Ci-dessous, l'heure est affichée toutes les secondes.

```
10 '---------- EVERY
20 MODE 1
30 hd=TIME                                ' heure depart
40 EVERY 50,0  GOSUB 90                   ' toutes les secondes
50 '
60 GOTO 60                                ' boucle d'attente
70 END
80 '--------------------- affichage heure
90 LOCATE 1,22:PRINT INT((TIME-hd)/300)
100 RETURN
```

Ci-dessous, un danseur est représenté dans quatre positions.

```
10 MODE 1
20 EVERY 60,0 GOSUB 140    ' 3 secondes horloge 0
30 EVERY 200,1 GOSUB 110   ' 10 secondes horloge 1
40 '
50 FOR c=248 TO 251        ' 4 Positions
60  LOCATE 20,15:PRINT CHR$(c)
70   SOUND 2,INT(RND*100)+50,20
80 NEXT c
90 GOTO 50
100 '---------- chrono 1
110 LOCATE 13,15:PRINT "smurf dance"
120 RETURN
130 '---------- chrono 0
140 LOCATE 13,15:PRINT "SMURF DANCE"
150 RETURN
```

Toutes les 3 secondes, un message est affiché en majuscules. Toutes les 10 secondes le même message est affiché en minuscules.

REMAIN (chrono)

Annule le chronomètre spécifié et donne le temps qui restait.

```
PRINT REMAIN(0)
```

DI

Annule les interruptions jusqu'à ce que l'instruction "EI" soit exécutée.

Ceci permet de "protéger" une partie de programme contre les interruptions intempestives qui provoqueraient par exemple une modification de la position du curseur graphique si le sous-programme d'interruption exécuté n'a pas prévu de restituer le curseur graphique dans l'état où il l'avait trouvé.

Si le programme principal et un sous-programme d'interruption utilisent une ressource commune (imprimante par exemple), DI permet d'éviter des conflits.

EI

Autorise les interruptions annulées par DI.

programme

LES FICHIERS SÉQUENTIELS | 12

- OPENOUT
- OPENIN
- PRINT # 9
- INPUT # 9
- EOF
- LINE INPUT # 9
- CLOSEOUT
- CLOSEIN
- LINE INPUT # 9
- WRITE # 9

Les instructions d'écriture et de lecture des fichiers séquentiels sur cassette sont analogues à celles de l'édition et de la lecture au clavier (PRINT et INPUT). Les informations transitent par une.mémoire tampon.

OPENOUT nom-fichier
OPENIN nom-fichier

Un fichier est ouvert en **écriture** par **OPENOUT** et en **lecture** par **OPENIN**.

PRINT # 9, variable
INPUT # 9, variable

PRINT # 9 écrit la variable spécifiée sur cassette.

Le programme ci-dessous écrit les deux variables NOM$ et TPH$ sur cassette en les séparant par des retour-chariot.

```
30 OPENOUT "FICH"
40 '
50 INPUT "Nom (ou FIN) ";nom$
60 IF nom$="FIN" THEN CLOSEOUT:GOTO 130
70 INPUT "Telephone ";tph$
80 '
90 PRINT #9,nom$
100 PRINT #9,tph$
110 GOTO 50
```

Pour lire les enregistrements, on écrit :

```
180 INPUT #9,nom$
190 INPUT #9,tph$
```

On peut également écrire : INPUT # 9,NOM$, TPH$.

ATTENTION :

```
10 PRINT #9,nom$
20 PRINT #9,tph$
```

ne doit pas être remplacé par : PRINT # 9, NOM$, TPH$; les valeurs seraient seulement séparées par des espaces et seraient considérées à la lecture comme une seule valeur.

DUPONT	R C	044.22.63	R C	MARTIN	R C	955.19.01

Avec INPUT # 9, une virgule dans une chaîne est considérée comme séparateur. Par conséquent, les chaînes comportant des virgules doivent être écrites par WRITE # 9 ou être lues par INPUT # 9 qui ne considère comme séparateur que le retour-chariot.

EOF

EOF teste la fin d'un fichier en lecture. Cette instruction doit être programmée **avant** la lecture par INPUT # 9.

```
160 IF EOF=-1 THEN CLOSEIN:END
170 '
180 INPUT #9,nom$
```

CLOSEOUT
CLOSEIN

CLOSEOUT provoque le transfert sur cassette de la mémoire tampon.

CLOSEIN ferme un fichier en lecture.

LINE INPUT # 9, chaîne

Lit une chaîne de caractères dans un fichier en ne considérant comme séparateur que le retour-chariot (code 13). La virgule n'est pas considérée comme séparateur.

WRITE # 9, variable1, variable2

Ecrit les valeurs en les séparant par des virgules. Les chaînes sont placées entre guillemets.

```
100 rue$="11,rue NOBEL":mt=164553
110 WRITE #9,rue$,mt
```

```
"11,rue NOBEL", 164 553
```

Les chaînes comportant des virgules peuvent ainsi être lues par INPUT # 9.

□ Le programme ci-dessous crée un **annuaire téléphonique** et le lit.

```
10 '-------------- fichier
20 '--------------- ecriture
30 OPENOUT "FICH"
40 '
50 INPUT "Nom (ou FIN) ";nom$
60 IF nom$="FIN" THEN CLOSEOUT:GOTO 130
70 INPUT "Telephone ";tph$
80 '
90 PRINT #9,nom$
100 PRINT #9,tph$
110 GOTO 50
120 '-------------- lecture
130 PRINT "Rembobinez"
140 OPENIN "FICH"
150 '
160 IF EOF=-1 THEN CLOSEIN:END        ' fin de fichier?
170 '
180 INPUT #9,nom$
190 INPUT #9,tph$
200 PRINT nom$,tph$
210 GOTO 160

Ready
run
Nom (ou FIN) ? DUPONT
Téléphone ? 044-22-63
Nom (ou FIN) ? MARTIN
Téléphone ? 999-88-77
NOM (ou FIN) ? FIN
Press REC and PLAY then any key :
Saving FICH block 1
Rembobinez
Press PLAY then any key :
Loading FICH block 1
DUPONT        044-22-63
MARTIN        999-88-77
```

□ Le programme ci-dessous sauvegarde une table sur cassette.

```
10 '--------------- SAUVEGARDE D'UNE TABLE
20 DIM A(200)
30 OPENOUT "TB"
40 '
50 FOR I=1 TO 200
60  A(I)=I
70  PRINT #9,A(I)
80 NEXT I
90 CLOSEOUT
100 '-------------------- lecture de la table
110 OPENIN "TB"
120 FOR I=1 TO 200
130  INPUT#9,A(I)
140  PRINT a(i)
150 NEXT I
160 CLOSEIN
```

Un programme sauvegardé en ASCII peut être lu comme un fichier séquentiel.

LES PROGRAMMES | 13

GRAPHISMES

- □ Tracé d'un dessin par segments de droites
- □ Dessinateur
- □ Tracé d'un dessin défini en DATA
- □ Géographie
- □ Tracé d'un dessin en relatif
- □ Rotation d'une figure
- □ Squash
- □ Tracé de courbe
- □ Histogramme
- □ Histogramme 3D

GESTION

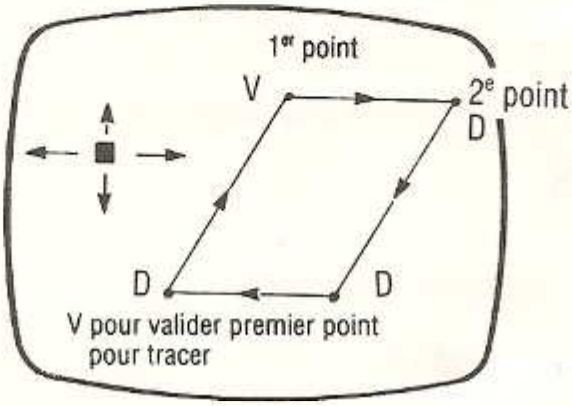
- □ Saisie d'écran
- □ Fichier d'adresses
- □ Gestion de fichier automatique
- □ Bibliothèque

GRAPHISMES

TRACÉ D'UN DESSIN PAR SEGMENTS DE DROITES ET DIGITALISATION D'UN DESSIN

Le programme ci-dessous permet de déplacer un curseur à l'aide de 4 flèches →← ↑ ↓. et de **tracer des droites** entre ces points.

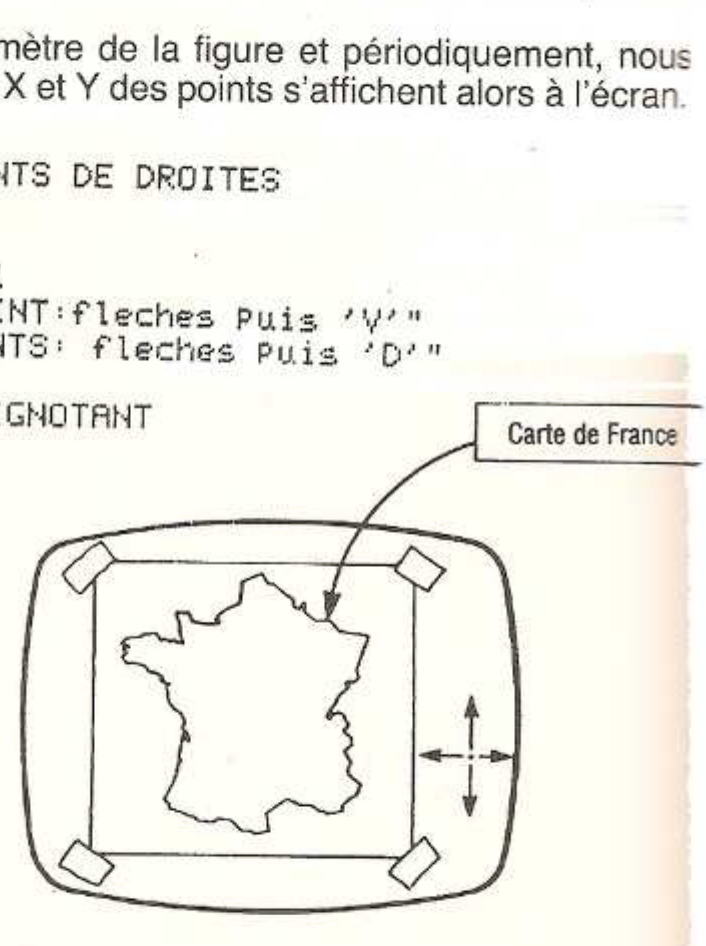
- **Pour "valider"** le premier point, nous appuyons sur "V".
- **Pour tracer une droite** entre le point courant et le point valide précédent, nous appuyons sur "D".



Le programme permet également de **"digitaliser"** un dessin :

- Nous collons sur l'écran une feuille transparente sur laquelle a été décalquée la carte de France par exemple.
- Nous déplaçons le curseur sur le périmètre de la figure et périodiquement, nous "validons" les points. Les coordonnées X et Y des points s'affichent alors à l'écran.

```
10 '------------ TRACE PAR SEGMENTS DE DROITES
20 ce=1:cf=0
30 MODE 1
40 INK 0,26:INK 1,0:PAPER 0:PEN 1
50 LOCATE 1,20:PRINT "PREMIER POINT:fleches Puis 'V'"
60 LOCATE 1,21:PRINT "AUTRES POINTS: fleches Puis 'D'"
70 x=200:y=200
80 '---------------- CURSEUR CLIGNOTANT
90 t=TEST(x,y)
100 '
110 c$=INKEY$:IF c$<>"" THEN 160
120 PLOT x,y,ce
130 PLOT x,y,cf
140 GOTO 110
150 '
160 PLOT x,y,t
170 c=ASC(c$)
175 c$=UPPER$(c$)
180 IF C=242 THEN X=X-2
190 IF C=243 THEN X=X+2
200 IF C=240 THEN Y=Y+2
```



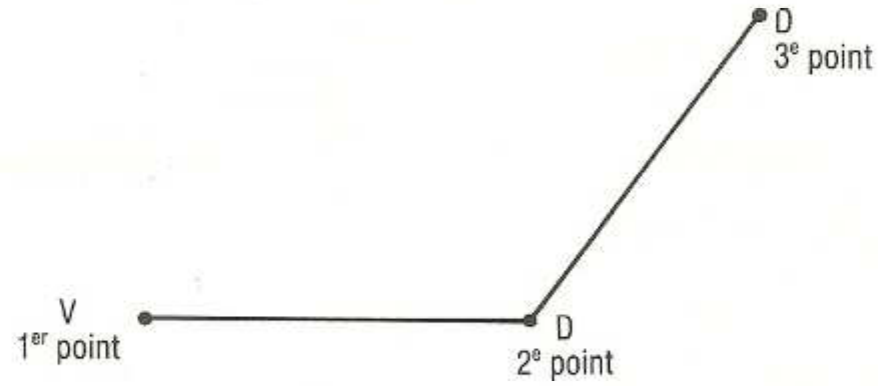
→

```
210 IF C=241 THEN Y=Y-2
220 LOCATE 1,17:PRINT c$
230 IF C$="V" THEN PLOT X,Y,CE:GOSUB 270:XA=X:YA=Y
240 IF C$="D" THEN PLOT XA,YA,CE:GOSUB 270:DRAW X,Y,CE:YA=Y:
XA=X
250 GOTO 90
260 '--------- AFFICHAGE X,Y
270 LOCATE 1,14:PRINT X;Y:RETURN
```

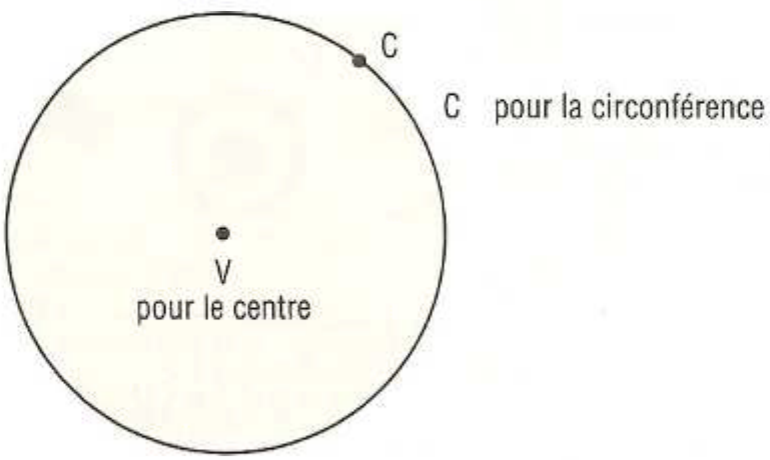
DESSINATEUR

Ce programme "dessinateur" permet de représenter des droites, des rectangles pleins, des cercles pleins ainsi que des triangles pleins.

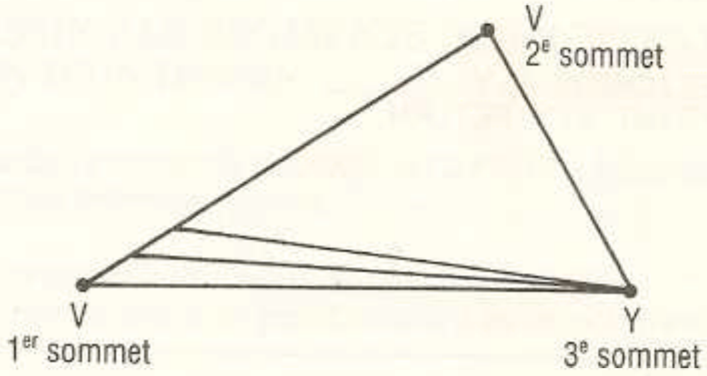
Pour dessiner une droite par exemple, il faut appuyer sur "V" pour valider le premier point puis déplacer le curseur avec les flèches (→, ←, ↓, ↑) et appuyer sur "D" pour obtenir le tracé entre les deux points.



Un cercle se représente en appuyant sur "V" pour définir le centre, puis en déplaçant le curseur jusqu'à un point de la circonférence et en appuyant sur "C" pour le valider. Le tracé du cercle est alors effectué.



Le triangle plein s'obtient en appuyant sur "V" pour définir les deux premiers sommets puis sur "Y" pour le troisième sommet.



La droite YV (1er sommet) pivote pour "remplir" le triangle. Ce triangle plein permet de "remplir" un certain nombre de figures, comme le montre l'exemple du cube, et remplace ainsi partiellement l'instruction "PAINT" qui existe sur d'autres matériels (MSX eg.).

En utilisant la couleur de fond "F" et "R", il est possible d'effacer des parties coloriées. Attention, le curseur disparaît sur une zone non coloriée ; pour le faire apparaître frappez 1 ;

Pour tracer des figures discontinues, utiliser "V".

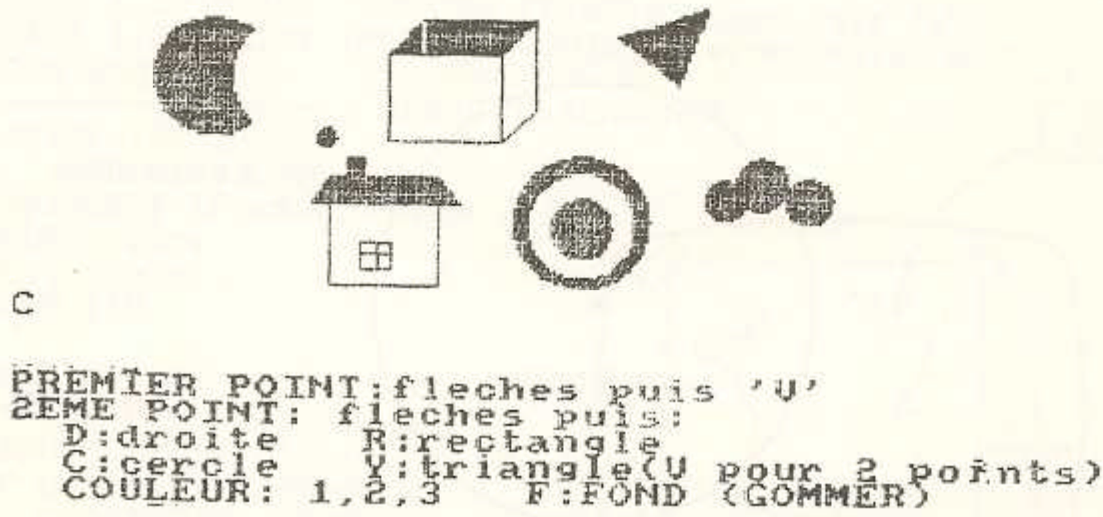
Pour sauvegarder l'écran graphique, faire :

SAVE "DES",B,&C000,&4000

LOAD "DES" permet de lire le dessin sauvegardé.

Ces instructions peuvent être insérées dans le programme :

```
355  IF C$ = "S" THEN SAVE...
356  IF C$ = "L" THEN LOAD...
```



Le signe "^" représente la flèche "↑".

```
10 '---------------- DESSINATEUR
20 ce=1:cf=0                          ' coul ecriture et fond
30 INK 0,26:INK 1,0:PAPER cf:PEN ce   ' fond blanc/encre noire
40 SPEED KEY 10,3                     ' repetition rapide
50 MODE 1
60 LOCATE 1,20:PRINT "PREMIER POINT:fleches puis 'V'"
70 LOCATE 1,21:PRINT "2EME POINT: fleches puis:"
80 LOCATE 3,22:PRINT "D:droite   R:rectangle"
90 LOCATE 3,23:PRINT "C:cercle   Y:triangle(V pour 2 points)"
100 LOCATE 3,24:PRINT "COULEUR: 1,2,3   F:FOND (GOMMER)"
110 LOCATE 3,25:PRINT "A:annulation derniere droite"
120 xa=200:ya=300                     ' point precedent
130 xb=xa:yb=ya:x=xa+20:y=ya
140 '----------------- CURSEUR CLIGNOTANT
150 t=TEST(x,y)
160 '
170 c$=INKEY$:IF c$<>"" THEN 210
180 PLOT x,y,ce:PLOT x,y,cf
190 GOTO 170
200 '
210 PLOT x,y,t
220 c=ASC(c$)
230 IF C=242 THEN IF x>2 THEN X=X-2:GOTO 150
240 IF C=243 THEN IF x<638 THEN X=X+2:GOTO 150
250 IF C=240 THEN IF y<398 THEN Y=Y+2:GOTO 150
260 IF C=241 THEN IF y>2 THEN Y=Y-2:GOTO 150
270 c$=UPPER$(c$):LOCATE 1,17:PRINT c$
280 IF C$="V" THEN PLOT x,y,ce:xb=xa:yb=ya:xa=x:ya=y
290 IF C$="D" THEN PLOT xa,ya,ce:DRAW x,y,ce:xb=xa:yb=ya:ya=y:
xa=x
300 IF C$="C" THEN GOSUB 430
310 IF c$="R" THEN GOSUB 380:xa=x:ya=y
320 IF c$="Y" THEN GOSUB 510:xb=xa:yb=ya:xa=x:ya=y
330 IF c$="A" THEN PLOT xb,yb:DRAW xa,ya,cf:x=xb:y=yb:xa=xb:ya
=yb
340 IF C$="F" THEN CE=0
350 IF C$>="0" AND C$<="9" THEN CE=VAL(C$)
360 GOTO 150
370 '------------- rectangle plein
380 FOR y1=ya TO y STEP SGN(y-ya)
390 PLOT xa,y1,ce:DRAW x,y1,ce
400 NEXT y1
410 RETURN
420 '---------------------- Cercle plein
430 r=SQR((xa-x)^2+(ya-y)^2):r2=r^2
440 FOR dx=-r TO r
450  dy=SQR(r2-(dx^2))
460  PLOT xa+dx,ya+dy,ce:DRAW xa+dx,ya-dy,ce
470 NEXT dx
480 RETURN
490 '---------------------- triangle plein
500 ' valider 2 points avec 'V' puis un troisieme avec 'Y'
510 d=SQR((yb-ya)^2+(xb-xa)^2)
520 IF d=0 THEN RETURN
530 cx=(xa-xb)/d:cy=(ya-yb)/d
540 FOR dd=0 TO d STEP 0.5
550  x3=xb+dd*cx:y3=yb+dd*cy
560  PLOT x,y,ce:DRAW x3,y3,ce
570 NEXT dd
580 RETURN
```

TRACÉ DE DESSIN DÉFINI EN DATA

Le programme ci-dessous trace un figure en reliant entre eux des points définis en DATA.

```
10 '------------------------- carte de france
20 MODE 2:INK 0,1:INK 1,24:PAPER 0:PEN 1
30 DATA 276,370
40 DATA 328,342,   398,318,   380,280
50 DATA 380,260,   364,240,   364,234
60 DATA 372,234,   376,240,   386,240
70 DATA 378,226,  384,204,   384,196
80 DATA 396,188,   406,186,  406,174
90 DATA 384,166,  366,156,  352,166
100 DATA 332,170,  318,164,  308,150
110 DATA 306,136,  300,132,  236,148
120 DATA 196,156,  208,188,  212,218
130 DATA 206,236,  188,250,  174,272
140 DATA 164,282,  146,290,  132,290
150 DATA 132,306,  168,310,   182,304
160 DATA 198,302,  186,314,   184,334
170 DATA 200,334,  204,328,   222,322
180 DATA 234,322,   238,330,  242,338
190 DATA 256,346,   262,362,  276,370
200 DATA 999,999
210 '
220 READ xa,ya:PLOT xa,ya
230 '
240 READ x,y
250 IF x=999 THEN END
260 '
270 DRAW x,y,1
280 GOTO 240
```

Pour tracer une figure discontinue, nous spécifions une coordonnée fictive égale à 0.0

```
50 MODE 1
60 DATA 100,100
70 DATA 150,100,  150,120
80 DATA 0,0
90 DATA 200,200
100 DATA 250,200,  250,240
110 DATA 999,999
120 '
130 READ xa,ya          ' Premier Point
140 PLOT xa,ya
150 '
160 READ x,y
170 IF x=999 THEN END
180 IF x=0 AND y=0 THEN READ xa,ya:PLOT xa,ya:GOTO 160
190 DRAW x,y
200 GOTO 160
```

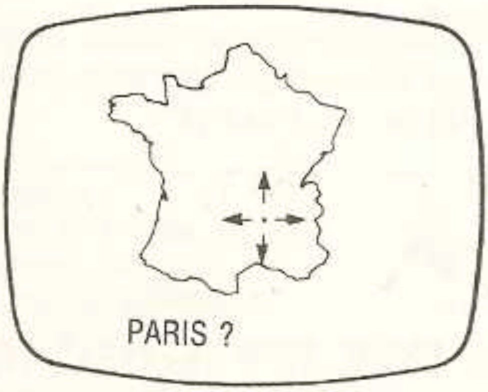
GÉOGRAPHIE

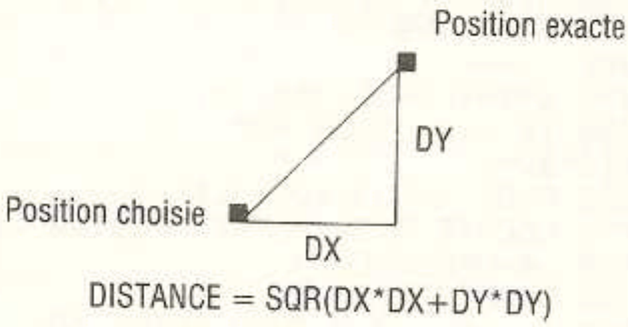
Nous proposons deux programmes de géographie.

- Le premier affiche la position d'une ville et demande le nom de la ville.



- Le second demande de situer la position d'une ville. L'élève doit déplacer le curseur à l'aide des quatre flèches et valider la position avec "V". Le programme calcule l'écart entre la position réelle et la position choisie.





Pour ce second programme, utiliser les lignes 10 à 400 du premier programme et ajouter les lignes 480 à 830 de la version 2.

```
10 '---------------------- INTERROGATION GEOGRAPHIE  NO 1
20 MODE 1
30 INK 0,4:INK 1,24:PAPER 0:PEN 1
40 DIM XV(30),YV(30),V$(30)
50 DATA 276,370
60 DATA 328,342,   398,318,   380,280
70 DATA 380,260,   364,240,   364,234
80 DATA 372,234,   376,240,   386,240
90 DATA 378,226,  384,204,   384,196
100 DATA 396,188,   406,186,  406,174
110 DATA 384,166,  366,156,  352,166
120 DATA 332,170,  318,164,  308,150
130 DATA 306,136,  300,132,  236,148
140 DATA 196,156,  208,188,  212,218
150 DATA 206,236,  188,250,  174,272
160 DATA 164,282,  146,290,  132,290
170 DATA 132,306,  168,310,   182,304
180 DATA 198,302,  186,314,   184,334
190 DATA 200,334,  204,328,   222,322
```



Quelle ville ?

```
200 DATA 234,322,   238,330,  242,338
210 DATA 256,346,   262,362,  276,370
220 DATA 999,999
230 '
240 DATA 276,302,PARIS
250 DATA 286,348,LILLE
260 DATA 314,314,REIMS
270 DATA 242,324,LE HAVRE
280 DATA 196,328,CHERBOURG
290 DATA 142,302,BREST
300 DATA 186,288,RENNES
310 DATA 186,264,NANTES
320 DATA 230,254,POITIERS
330 DATA 214,206,BORDEAUX
340 DATA 336,214,LYON
350 DATA 334,180,AVIGNON
360 DATA 358,168,MARSEILLE
370 DATA 999,999,ZZZZ
380 '
390 READ xa,ya:PLOT xa,ya,1
400 '
410 READ x,y:IF x=999 THEN 450
420 DRAW x,y,1
430 GOTO 410
440 '--
450 READ x,y,v$:IF x=999 THEN 490
460 nv=nv+1:xv(nv)=x:yv(nv)=y:v$(nv)=v$
470 GOTO 450
480 '----
490 v=INT(RND(1)*nv)+1                ' ville au hasard
500 IF v=av THEN 490
510 av=v
520 PLOT xv(v),yv(v),1
530 LOCATE 5,20:INPUT "Quelle ville ";v$
540 v$=UPPER$(v$)
550 LOCATE 5,21
560 IF v$(v)=v$ THEN PRINT "OK " ELSE PRINT "NON ,C'est ";v$(v
)
570 FOR tp=1 TO 2000:NEXT tp
580 PLOT xv(v),yv(v),0
590 LOCATE 5,21:PRINT SPC(30)
600 LOCATE 5,20:PRINT SPC(30)
610 GOTO 490
```

```
480 '----
485 ' ajouter les instructions 10-470 du programme precedent
490 x=276:y=302
500 SPEED KEY 10,2
510 '
520 v=INT(RND(1)*nv)+1                ' ville au hasard
530 IF v=av THEN 520 ELSE av=v
540 LOCATE 5,20:PRINT "OU EST SITUE ";V$(V)
550 '-----------------------gestion curseur
560 LOCATE 5,22:PRINT "Fleches puis 'V'"
570 '
580 t=TEST(x,y)
590 '
600 c$=INKEY$:IF c$<>""  THEN 650
610 PLOT x,y,1
```

→

```
620 PLOT x,y,0
630 GOTO 600
640 '
650 PLOT x,y,t
660 '
670 c=ASC(c$)
680 IF c=242 THEN x=x-2
690 IF c=243 THEN x=x+2
700 IF c=240 THEN y=y+2
710 IF c=241 THEN y=y-2
720 '
730 IF C$="V" OR C$="v" THEN 760
740 GOTO 580
750 '------------- calcul distance
760 dx=xv(v)-x:dy=yv(v)-y
770 d=SQR(dx*dx+dy*dy)
780 LOCATE 5,23:PRINT "Vous etes a:";d*5 "Km";SPC(10)
790 PLOT xv(v),yv(v),1:PLOT xv(v)+2,yv(v),1
800 FOR tp=1 TO 2000:NEXT tp
810 LOCATE 5,20:PRINT SPC(30)
820 PLOT xv(v),yv(v),0:PLOT xv(v)+2,yv(v),0
830 GOTO 520
```

TRACÉ D'UN DESSIN EN RELATIF

Une figure définie en coordonnées relatives (chaque point est défini par rapport au précédent) peut être représentée facilement avec une échelle.

```
10 '--------------------- trace de la carte de FRANCE en relatif
20 MODE 1
30 INK 0,1:INK 1,24
40 ech=0.5                  ' echelle
50 xa=200:ya=200            ' 1er Point
60 '
70 DATA 52,-28,   70,-24,   -18,-38
80 DATA 0,-20,   -16,-20,    0,-6
90 DATA 8,0,      4,6,      10,0
100 DATA -8,-14,  6,-22,     0,-8
110 DATA 12,-8,   10,-2,     0,-12
120 DATA -22,-8, -18,-10,   -14,10
130 DATA -20,4,  -14,-6,   -10,-14
140 DATA -2,-14,  -6,-4,    -64,16
150 DATA -40,8,   12,32,     4,30
160 DATA  -6,18, -18,14,    -14,22
170 DATA -10,10, -18,8,    -14,0
180 DATA 0,16,    36,4,     14, -6
190 DATA 16,-2,  -12,12,    -2,20
200 DATA 16,0,    4,-6,     18,-6
210 DATA 12,0,    4,8,      4,8
220 DATA 14,8,    6,16,     14,8
230 DATA 999,999
240 '
250 PLOT xa,ya
260 '
270 READ dx,dy:IF dx=999 THEN END
280 DRAWR ech*dx,ech*dy,1
290 GOTO 270
```

Le programme ci-dessous convertit les coordonnées absolues en coordonnées relatives.

```
10 '----------------------- conversion absolu-->relatif
20 DATA 276,370
30 DATA 328,342,   398,318,   380,280
40 DATA 380,260,   364,240,   364,234
50 DATA 372,234,   376,240,   386,240
60 ' suite carte de FRANCE
70 '
190 DATA 999,999
200 '
210 READ xa,ya          ' Premier Point
220 '-
230 READ x,y
240 IF x=999 THEN END
250 dx=x-xa:dy=y-ya
260 PRINT #8,dx,dy      ' imprimante
270 xa=x:ya=y
280 GOTO 230

 52           -28
 70           -24
-18           -38
 0            -20
-16           -20
 0            -6
 8             0
 4             6
 10            0
```

ROTATION D'UNE FIGURE

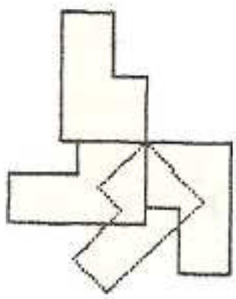
Ce programme représente une figure définie dans la position demandée.

```
10 '------------------------- rotation d'une figure
20 '
30 MODE 1
40 nP=7                    ' nombre de Points
50 DATA 200,200
60 DATA 250,200
70 DATA 250,120
80 DATA 220,120
90 DATA 220,160
100 DATA 200,160
110 DATA 200,200
120 '
130 FOR i=1 TO nP:READ x(i),y(i):NEXT i
140 '
150 x0=x(1):y0=y(1)                ' centre de rotation
160 '
170 LOCATE 1,2:INPUT "Quel angle(degre) ";a
180 a=a*2*PI/360
190 s=SIN(a):c=COS(a)
200 FOR P=1 TO nP
210  dx=x(P)-x0:dy=y(P)-y0
220  x1(P)=x0+dx*c+dy*s
230  y1(P)=y0+dy*c-dx*s
```

→

```
240 NEXT p
250 ce=1:GOSUB 280
260 GOTO 170
270 '---------------------------- spgm trace
280 PLOT x1(1),y1(1),ce
290 FOR p=2 TO np
300  DRAW x1(p),y1(p),ce
310 NEXT p
320 RETURN
```

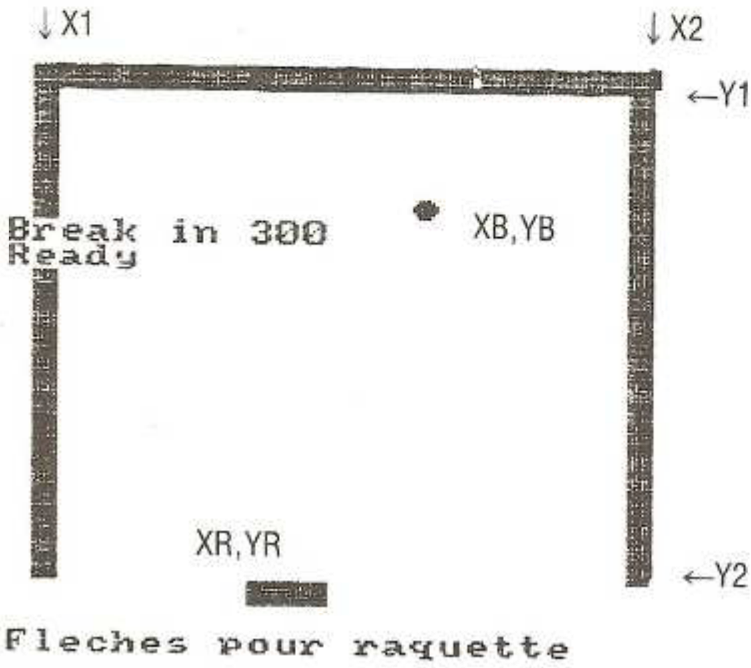
```
Quel angle(degre) ? *Break*
Break in 170
Ready
```



SQUASH

Vous devez faire rebondir une balle à l'aide d'une raquette que vous déplacez avec les deux flèches → et ←.

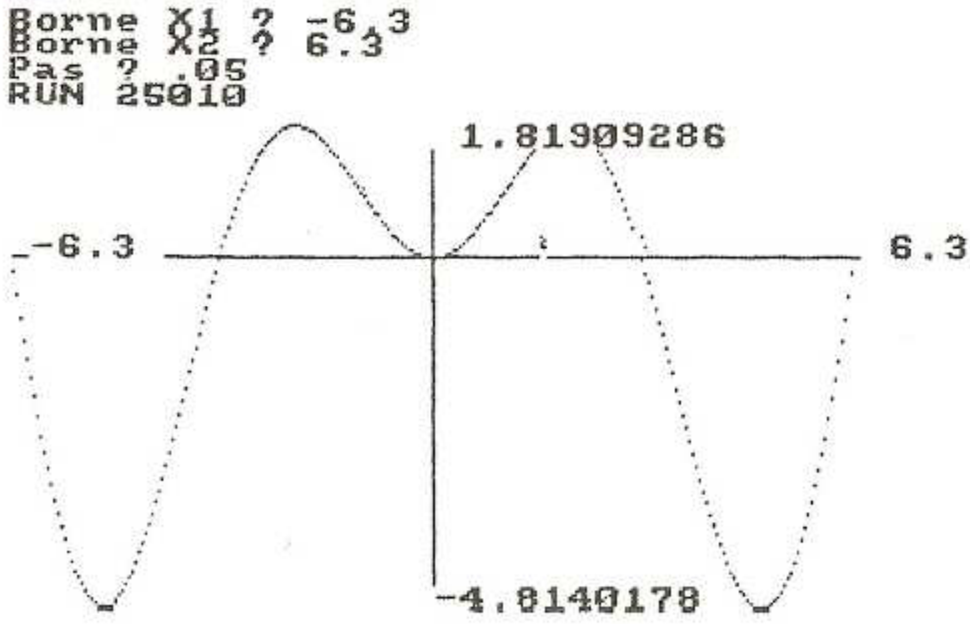
Nous avons utilisé la fonction INKEY(n° touche) de façon à obtenir un déplacement rapide de la raquette. Avec INKEY$, le délai de répétition des touches devrait être modifié par SPEED KEY.



```
10 '-------------- SQUASH
20 CLS:INPUT "Niveau (1,2,3) ";nv
30 '----------- dessin terrain
40 MODE 1
50 x2=18+nv*2
60 x1=2:y1=2:y2=22
70 CLS
80 LOCATE 1,24:PRINT "Fleches pour raquette"
90 FOR x=x1 TO x2:LOCATE x,y1:PRINT CHR$(143):NEXT x
100 FOR y=y1 TO y2-1:LOCATE x1,y:PRINT CHR$(143):LOCATE x2,y:P
RINT CHR$(143):NEXT y
110 '
120 rb=0                          ' rebonds
130 rq$=CHR$(32)+CHR$(143)+CHR$(143)+CHR$(143)+CHR$(32)
140 dx=1:dy=-1
150 xb=5+INT(RND(1)*5):yb=10      ' balle
160 xr=10:yr=y2                   ' raquette
170 LOCATE xr-1,yr:PRINT rq$
180 '------------------------ deplacement balle
190 LOCATE xb,yb:PRINT CHR$(32)  ' effacement balle
200 xb=xb+dx:yb=yb+dy             ' nouvelle position
210 IF yb=y1+3 THEN  yb=yb+INT(RND*2)*dy
220 LOCATE xb,yb:PRINT CHR$(231)
230 IF xb>=x2-1 THEN dx=-dx       ' rebonds
240 IF yb<y1+2 THEN dy=-dy
250 IF xb<x1+2 THEN dx=-dx
260 '
270 IF xb>xr-2 AND xb<xr+4 AND yb>yr-2 THEN dy=-dy:rb=rb+1
280 '
290 IF yb>=YR THEN LOCATE 30,23:PRINT rb;"POINTS":GOTO 380
300 '--------------------- deplacement raquette
310 FOR tp=1 TO 10:NEXT tp
320 '
330 IF INKEY(1)=0 THEN IF xr<x2-3 THEN xr=xr+1
340 IF INKEY(8)=0 THEN IF xr>x1+1 THEN xr=xr-1
350 LOCATE xr-1,yr:PRINT rq$
360 GOTO 190
370 '
380 FOR TP=1 TO 2000:NEXT TP:GOTO 50
```

TRACÉ DE COURBE

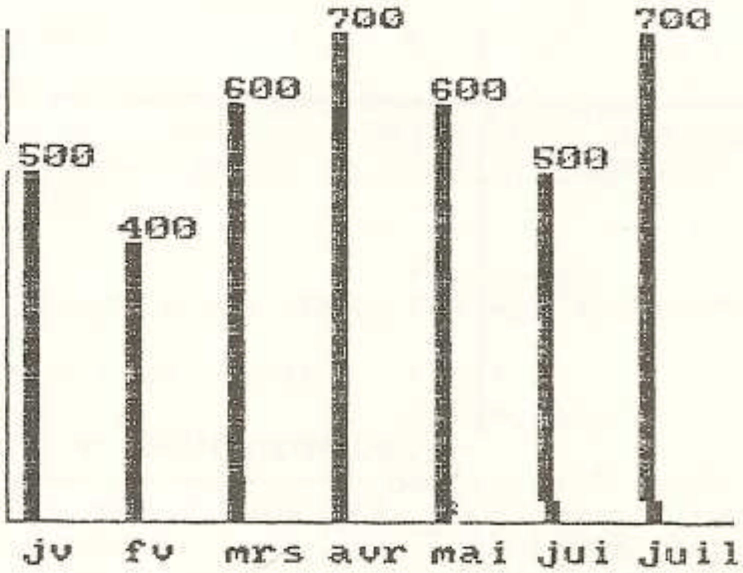
Ce programme trace la courbe d'une fonction écrite en 400. Les échelles sont calculées par le programme.




```
10 '------------- trace de courbe
20 MODE 2:PAPER 0:PEN 1
30 INK 0,1:INK 1,24
40 hecr=300:lecr=500        ' hauteur /largeur ecran
50 INPUT "Borne X1 ";b1
60 INPUT "Borne X2 ";b2
70 INPUT "Pas ";Pas:IF Pas=0 THEN Pas=0.1
80 '---------------- recherche mini/maxi
90 x=b1:GOSUB 400:y1=y:y2=y
100 FOR x=b1 TO b2 STEP Pas
110  GOSUB 400
120  IF y<y1 THEN y1=y
130  IF y>y2 THEN y2=y
140 NEXT x
150 ex=lecr/(b2-b1)      ' echelle x
160 ey=(hecr)/(y2-y1)
170 '------------ axe y
180 IF b2>=0 AND b1<=0 THEN x=-ex*b1:PLOT x,1:DRAW x,hecr,1
190 '-------------- axe x
200 IF y2>=0 AND y1<=0  THEN y=-y1*ey:PLOT 1,y:DRAW lecr,y
210 '------------------- courbe
220 FOR x=b1 TO b2 STEP Pas
230  GOSUB 400
240  sx=(x-b1)*ex:sy=(y-y1)*ey
250  PLOT sx,sy,1
260 NEXT x
270 '------------------ affichage extremes
280 y=-y1*ey+12:x=12:b=b1:GOSUB 340
290 y=-y1*ey+12:x=lecr:b=b2:GOSUB 340
300 y=12:x=ABS(b1)*ex:b=y1:GOSUB 340
310 y=hecr:x=ABS(b1)*ex:b=y2:GOSUB 340
320 END
330 '--
340 IF x>lecr THEN x=lecr/2
350 IF y>hecr THEN y=hecr/2
360 PLOT x,y
370 TAG:PRINT B;
380 RETURN
390 '----------------- courbe a tracer
400 Y=SIN(X)*X
410 RETURN
```

HISTOGRAMME

Ce programme d'histogramme calcule l'échelle automatiquement.




```
10 '------------------------- HISTOGRAMME
20 MODE 2
30 INK 0,1:INK 1,24:PAPER 0:PEN 1
40 nm=7                               ' nombre de mois
50 DATA jv,500,  fv,400
60 DATA mrs,600, avr,700
70 DATA mai,600, jui,500
80 DATA juil,700
90 '--
100 FOR i=1 TO nm:READ mois$(i),vnte(i):NEXT i
110 '
120 xa=20:ya=30                        ' depart axes
130 ix=50                              ' intervalle X
140 hecr=300                           ' hauteur ecran
150 '------------------------------ recherche maxi
160 mx=vnte(1)
170 FOR m=2 TO nm
180  IF vnte(m)>mx THEN mx=vnte(m)
190 NEXT m
200 ech=hecr/mx                        ' echelle
210 '------------------------------ axes
220 PLOT xa,ya:DRAW xa+nm*ix,ya,1
230 PLOT xa,ya:DRAW xa,hecr+ya,1
240 '-------------------------------- affichage mois
250 FOR m=1 TO nm
260  x=xa+8+ix*(m-1):y=ya-12
270  PLOT x,y:TAG:PRINT mois$(m);
280 NEXT m
290 '------------------------------- courbe
300 FOR m=1 TO nm
310  x1=xa+10+ix*(m-1):y1=ya+vnte(m)*ech
320  FOR dx=0 TO 5
330   PLOT x1+dx,ya:DRAW x1+dx,y1,1
340  NEXT dx
350 NEXT m
360 '------------------------------ affichage valeurs
```

→

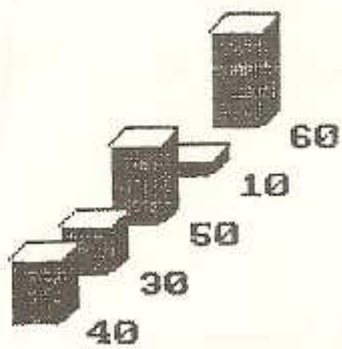
```
370 FOR m=1 TO nm
380   y=ya+vnte(m)*ech+16
390   x=xa+ix*(m-1)-10
400   PLOT x,y:TAG:PRINT vnte(m);
410 NEXT m
```

HISTOGRAMME 3D

Ce programme d'histogramme en trois dimensions, écrit sur AMSTRAD, s'adapte sur tout matériel disposant du graphique haute résolution.

Pour un matériel disposant de l'instruction BOXF (boîte pleine), tel que le MO5 ou MSX, le programme devient plus simple.

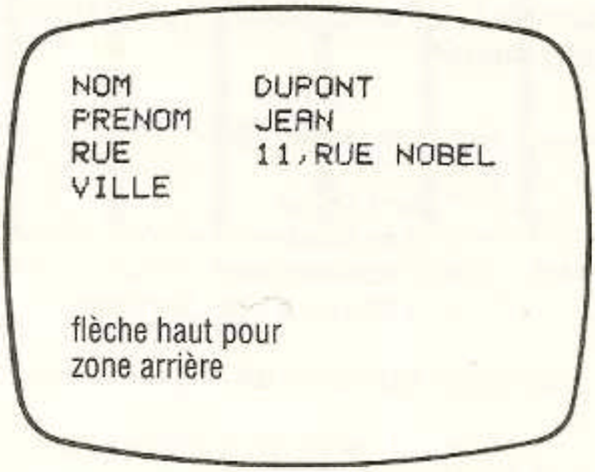
```
10 '---------------- histogramme 3D
20 MODE 2
30 xd=50:yd=50            ' origine
40 lg=30                  ' largeur
50 pr=10                  ' profondeur
60 itv=30                 ' intervalle batons
70 '
80 nh=5                   ' nombre de batons
90 '
100 h(1)=40
110 h(2)=30
120 h(3)=50
130 h(4)=10
140 h(5)=60
150 '----
160 FOR h=nh TO 1 STEP-1
170  xb=xd+(h-1)*itv                   ' origine baton
180  yb=yd+(h-1)*itv
190  FOR dx=1 TO pr                    ' 1 baton
200   PLOT xb+dx,yb+dx+h(h),1
210   IF dx>1 THEN PLOT xb+dx+lg,yb+dx,1:DRAW xb+dx+lg,yb+dx+h
(h),1:GOTO 250
220   FOR d1=1 TO lg                   ' rectangle
230    PLOT xb+dx+d1,yb+dx,1:DRAW xb+dx+d1,yb+dx+h(h),1
240   NEXT d1
250   PLOT xb+dx+1,yb+dx+h(h):DRAW xb+dx+lg,yb+dx+h(h),0
260  NEXT dx
270  PLOT xb+dx-1,yb+dx+h(h),1:DRAW xb+dx-1+lg,yb+dx+h(h),1
280  TAG:PLOT xb+30,yb:PRINT h(h);
290 NEXT h
```



GESTION

SAISIE D'ÉCRAN

Une saisie d'informations avec l'instruction "INPUT" ne permet pas de positionner sur une zone qui aurait été mal documentée. Le programme suivant le permet. Il utilise la fonction INKEY$. En outre, les caractères frappés au clavier peuvent être contrôlés dès leur introduction, sans attendre la frappe de ENTER.



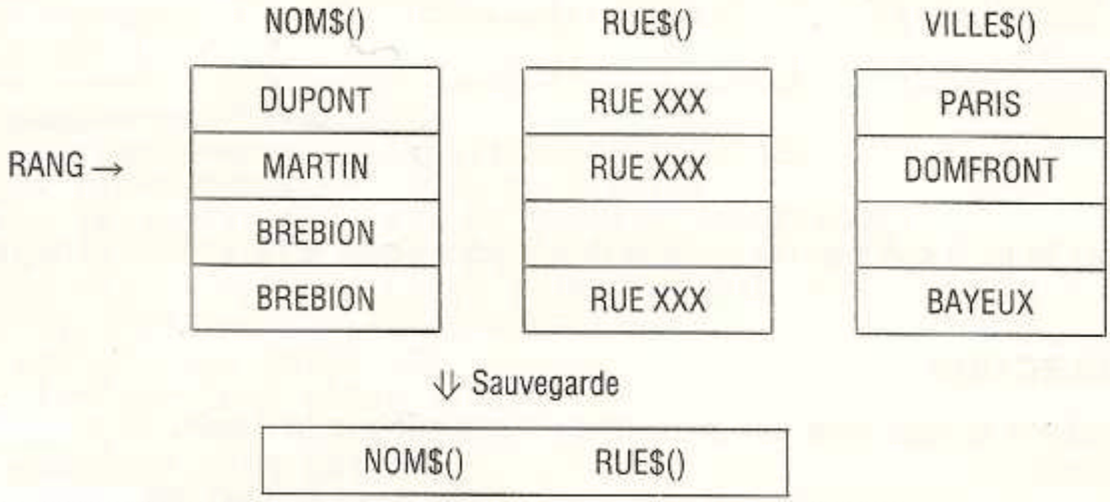

```
10 '-------------------- saisie dans une table FICH$( )
20 MODE 2
30 nrub=4                        ' nombre de rubriques
40 nrub$(1)="NOM"
50 nrub$(2)="PRENOM"
60 nrub$(3)="RUE"
70 nrub$(4)="VILLE"
80 '
90 GOSUB 130
100 PRINT:FOR ln=1 TO nrub:PRINT fich$(ln):NEXT ln
110 END
120 '---------------------- sous Programme saisie
130 CLS
140 LOCATE 1,20:PRINT "fleche haut Pour zone arriere"
150 FOR ln=1 TO nrub
160  LOCATE 1,ln+1:PRINT nrub$(ln)
170  LOCATE 10,ln+1:PRINT fich$(ln)        ' ancienne valeur
180 NEXT ln
190 '
200 FOR ln=1 TO nrub
210  xs=10:ys=ln+1
220  GOSUB 320
230  IF r=1 THEN fich$(ln)=lig$
240  IF r=3 THEN IF ln>1 THEN LOCATE xs,ys:PRINT fich$(ln):ln=
ln-1:GOTO 210 ELSE 210
250  LOCATE xs,ys:PRINT fich$(ln);SPC(20-LEN(fich$(ln)))
260 NEXT ln
270 PRINT:FOR ln=1 TO nrub:PRINT fich$(ln):NEXT ln    ' Pour te
st
```

→

```
280 END
290 '---------------------------- spgm saisie dans lig$
300 '    R=3 zone arriere
310 '
320 lig$=""
330 '
340 lg=LEN(lig$):LOCATE xs+lg,ys:PRINT CHR$(143) :' 143:curseu
r
350 LOCATE xs+lg,ys
360 '
370 c$=INKEY$:IF c$="" THEN 370              ' attente frappe
380 c=ASC(C$)
390 '
400 IF c<>127 THEN 430                       ' code suppression
410 IF lg>0 THEN lig$=LEFT$(lig$,lg-1):PRINT CHR$(32):GOTO 340
 ELSE 340
420 '
430 IF c=13 THEN 500                         ' code ENTER?
440 IF c=240 THEN r=3:LOCATE xs+lg,ys:PRINT CHR$(32):RETURN
450 IF c<32 OR c>127 THEN PRINT CHR$(7);:GOTO 370
460 lig$=lig$+c$                             ' ajout caractere fr
appe
470 PRINT c$                                 ' affichage caracter
e frappe
480 GOTO 340
490 '
500 LOCATE xs+lg,ys:PRINT CHR$(32)
510 IF lig$<>"" THEN r=1 ELSE r=2
520 RETURN
```

FICHIER D'ADRESSES

Le programme ci-dessous permet d'introduire et de modifier des données indépendantes du programme. Elles sont temporairement stockées dans des tables qui sont ensuite sauvegardées sur cassette.



La variable "RANG" donne l'adresse de rangement dans les tables.

Le mode "C" permet à la fois de créer et de modifier des fiches.

En mode modification, la valeur de chaque zone est affichée puis le programme attend une nouvelle valeur. Si vous ne voulez pas modifier une zone, appuyez sur "ENTER" sans entrer de valeur.

L'utilisation de l'instruction LINE INPUT (au lieu de INPUT) permet d'introduire le caractère virgule dans les zones.

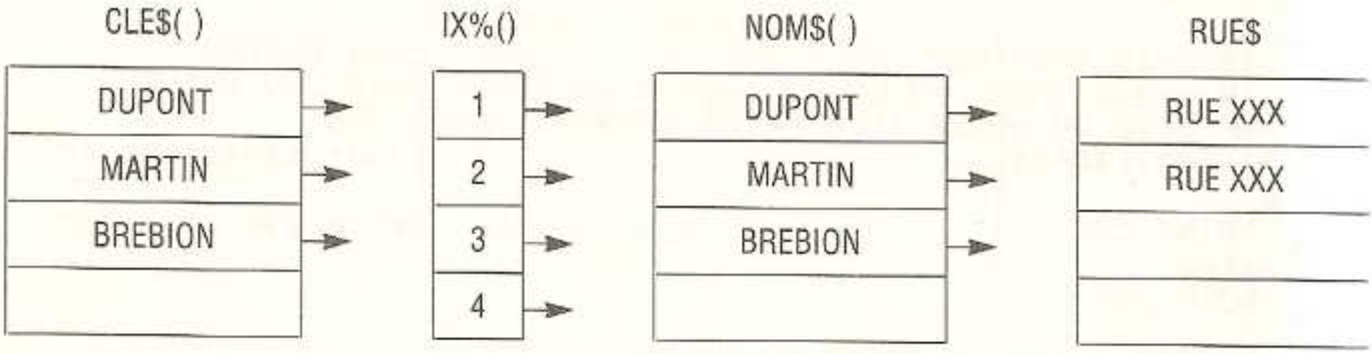
L'instruction 300 peut être remplacée par :

```
300 IF NOM$=LEFT$(NOM$(RANG),LEN(NOM$)) THEN 380
```

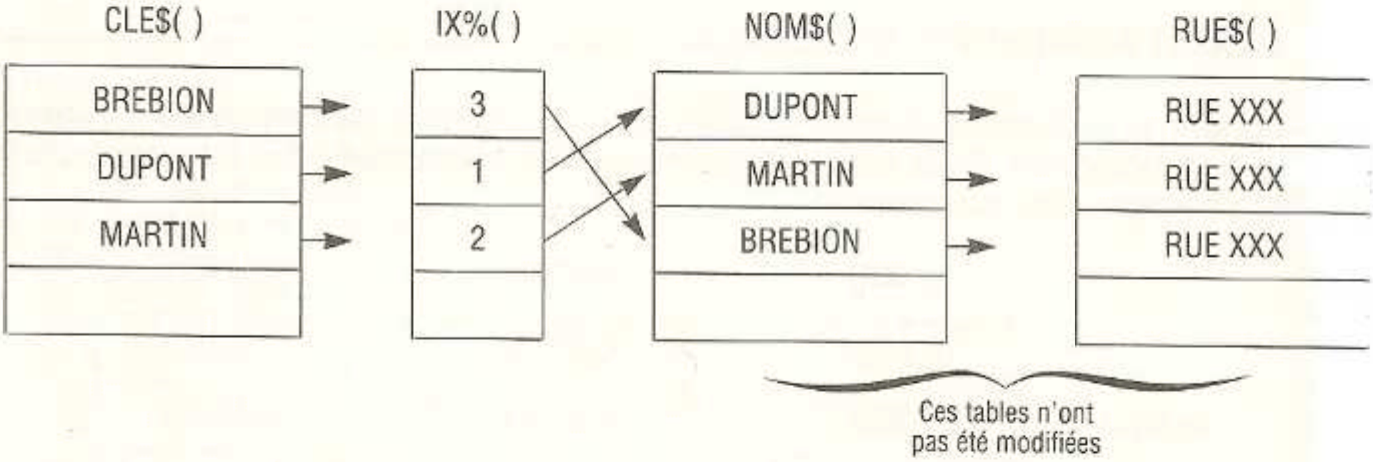
On peut ainsi entrer seulement les premières lettres du nom.

Pour obtenir une liste **triée par noms**, nous remplissons une "**table des Clés"** CLE$() avec les clés à trier.

Dans une **"table d'index"** IX%() nous rangeons les numéros de ligne (1, 2, 3...).



□ Nous trions les tables CLE$() et IX%()



□ Après le tri, il suffit de lire la table IX%() pour éditer les adresses dans l'ordre des noms.

TRI-SÉLECTION

Pour obtenir la liste triée des personnes d'une seule ville il suffit de sélectionner les noms ainsi :

```
1140  INPUT « Quelle ville », V$
1180  IF V$<> VILLE$(F) THEN 1200
```

TRI-MULTICRITÈRES

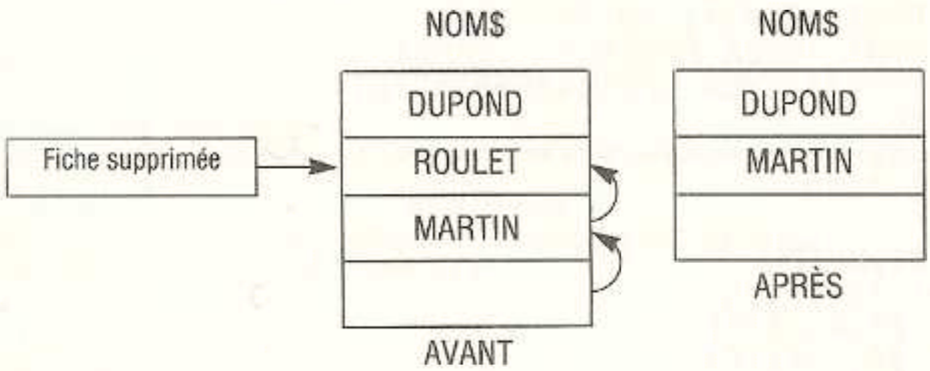
La liste alphabétique des clients dans l'ordre des villes s'obtient en faisant :
CLE$(F) = VILLE$(F) + NOM$(F) au lieu de CLE$(F) = NOM$(F).

CLE$ ( )

BOULOGNE MARTIN
BOULOGNE VINCENT
PARIS DUPONT
PARIS DURAND

Suppression :

Pour supprimer une fiche, nous décalons toutes les fiches en aval de la fiche supprimée.



Si vous n'effectuez qu'une sauvegarde, faire :

```
160 IF M$="FIN" THEN GOSUB 500:END
```

```
10 '-------------------------- fichier d'adresses
20 MODE 2:PRINT "CLAVIER MAJUSCULE"
30 DIM NOM$(100),RUE$(100),VILLE$(100),CPST$(100)
40 DIM CLE$(100),IX%(100)
50 NFICH=0                               ' nombre de fiches
60 INPUT "NOUVEAU FICHIER (O/N)";R$:IF R$="O" OR R$="o" THEN 90
70 GOSUB 680
80 '
90 CLS:PRINT "MODES:":PRINT
100 PRINT TAB(3) "C:     CREATION/MODIFICATION"
110 PRINT TAB(3) "LF:    LISTE DU FICHIER"
120 PRINT TAB(3) "FIN:   FIN DE SESSION(SAUVEGARDE)"
130 PRINT TAB(3) "LFN:   LISTE DU FICHIER PAR NOM"
135 PRINT TAB(3) "LFV:   LISTE TRIEE PAR VILLES"
140 PRINT:INPUT "MODE ";M$:M$=UPPER$(M$)
150 IF M$="C" THEN GOSUB 230
160 IF M$="FIN" THEN GOSUB 500:END
170 IF M$="LF" THEN GOSUB 810
180 IF M$="LFN" THEN GOSUB 930
190 IF M$="LFV" THEN GOSUB 1210
200 IF M$="S" THEN GOSUB 1450
210 GOTO 90
220 '================================ creation/modification
230 PRINT
240 PRINT "<ENTER> pour fin de mode"
250 PRINT "<ENTER> pour zone inchangee(en modif)":PRINT
260 LINE INPUT "NOM (ou <ENTER>) ? ",NOM$:IF NOM$="" THEN RETURN
270 '                        pas d'espace entre "" de IF NOM$=""
280 IF NFICH=0 THEN 330
```

→

```
290 FOR RANG=1 TO NFICH
300  IF NOM$=NOM$(RANG) THEN 380        nom existe t-il?
310 NEXT RANG
320 '---------------- nouveau nom
330 PRINT :INPUT "NOUVEAU NOM OK (O/N) ";R$:IF R$<>"O" AND R$<>"
o" THEN 230
340 NFICH=NFICH+1
350 RANG=NFICH
360 NOM$(RANG)=NOM$
370 '--------------------- entree/modification zones
380 PRINT
390 PRINT RUE$(RANG);TAB(15);                      ' ancienne valeur
400 LINE INPUT "RUE? ",RUE$:IF RUE$<>"" THEN RUE$(RANG)=RUE$
410 PRINT VILLE$(RANG);TAB(15);
420 LINE INPUT "VILLE? ",VILLE$
430 IF VILLE$<>"" THEN VILLE$(RANG)=VILLE$
440 PRINT CPST$(RANG);TAB(15);
450 LINE INPUT "CODE POSTAL? ",CPST$
460 IF CPST$<>"" THEN CPST$(RANG)=CPST$
470 GOTO 230
490 '================================ sauvegarde
500 OPENOUT "ADR"
510 '
520 PRINT #9,NFICH
530 FOR F=1 TO NFICH
540  PRINT #9,NOM$(F)
550  PRINT #9,RUE$(F)
560  PRINT #9,VILLE$(F)
570  PRINT #9,CPST$(F)
580 NEXT F
590 CLOSEOUT
600 RETURN
670 '================================ lecture fichier
680 OPENIN "ADR"
690 '
700 INPUT #9,NFICH
710 FOR F=1 TO NFICH
720  LINE INPUT #9,NOM$(F)
730  LINE INPUT #9,RUE$(F)
740  LINE INPUT #9,VILLE$(F)
750  LINE INPUT #9,CPST$(F)
760 NEXT F
770 CLOSEIN
780 PRINT:PRINT NFICH;"FICHES"
790 FOR TP=1 TO 2000:NEXT TP
800 RETURN
810 '================================ liste du fichier
820 PRINT "LISTE DU FICHIER":PRINT
830 '
840 FOR F=1 TO NFICH
850  IF F MOD 20=0 THEN INPUT "APPUYER SUR ENTER";X$
860  PRINT NOM$(F);TAB(13);
870  PRINT VILLE$(F);
880  PRINT
890 NEXT F
900 PRINT:INPUT "APPUYER SUR <ENTER> ";X$
910 RETURN
920 '================================ liste triee par nom
```

```
930 FOR F=1 TO NFICH
940  CLE$(F)=NOM$(F):IX%(F)=F
950 NEXT F
960 '
970 NC=NFICH                         ' nc: nombre de cles
980 GOSUB 1090                        ' appel tri
990 '
1000 CLS:PRINT "LISTE TRIEE PAR NOM":PRINT
1010 FOR F=1 TO NC
1020  X=IX%(F)
1030  PRINT NOM$(X);TAB(15);
1040  PRINT VILLE$(X)
1050 NEXT F
1060 PRINT:INPUT "APPUYER SUR <ENTER> ";X$
1070 RETURN
1080 '-------------------------- tri shell
1090 ECART=NC
1100 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
1110 IV=0
1120 FOR K=1 TO NC-ECART
1130  J=K+ECART
1140  IF CLE$(J)>=CLE$(K) THEN 1170
1150  X$=CLE$(K):CLE$(K)=CLE$(J):CLE$(J)=X$:IV=1
1160  X=IX%(K):IX%(K)=IX%(J):IX%(J)=X
1170 NEXT K
1180 IF IV=1 THEN 1110
1190 GOTO 1100
1200 '============================= liste triee par ville
1210 INPUT "VILLE (ENTER pour toutes) ";CLE$
1220 NC=0                             ' nombre de cles
1230 LG=LEN(CLE$)
1240 FOR F=1 TO NFICH
1250  IF CLE$<>LEFT$(VILLE$(F),LG) THEN 1270
1260  NC=NC+1:CLE$(NC)=VILLE$(F):IX%(NC)=F
1270 NEXT F
1340 GOSUB 1090                       ' appel tri
1350 CLS:PRINT "LISTE TRIEE PAR VILLE":PRINT
1360 FOR F=1 TO NC
1370  IF F MOD 20=0 THEN INPUT "APPUYER SUR ENTER";X$
1380  X=IX%(F)
1390  PRINT VILLE$(X);TAB(15);
1400  PRINT NOM$(X)
1410 NEXT F
1420 PRINT :INPUT "APPUYER SUR <ENTER> ";X$
1430 RETURN
1440 '================================ suppression
1450 PRINT:LINE INPUT "NOM? ";NOM$:IF NOM$="" THEN RETURN
1460 '
1470 FOR RANG=1 TO NFICH
1480  IF NOM$(RANG)=NOM$ THEN 1520
1490 NEXT RANG
1500 PRINT:PRINT "N'existe pas":PRINT:GOTO 1450
1510 '
1520 PRINT
1530 INPUT "SUPPRESSION OK (O/N) ";R$:IF R$<>"O" THEN 1450
1540 FOR J=RANG TO NFICH-1
1550  NOM$(J)=NOM$(J+1)
1560  RUE$(J)=RUE$(J+1)
1570  VILLE$(J)=VILLE$(J+1)
```

→

```
1580  CPST$(J)=CPST$(J+1)
1590 NEXT J
1600 NOM$(NFICH)="":RUE$(NFICH)="":VILLE$(NFICH)="":CPST$(NFICH)
=""
1610 NFICH=NFICH-1
1620 GOTO 1450
```

```
Mode ? C

<ENTER> pour fin de mode
<ENTER> pour zone inchangee(modif)

Nom (ou <ENTER>? DURAND

Nouveau nom OK (O/N) ? O

              Rue? 13,RUE DE MILAN
              Ville? PARIS
              Code postal? 75000

<ENTER> pour fin de mode
<ENTER> pour zone inchangee(modif)

Nom (ou <ENTER>?
Break in 260
Ok
```

```
Liste triee par nom

BALU          RUE XXX        ST CLOUD
BESSE         RUE DE MILAN   TOULON
DUPONT        11,RUE NOBEL   MONTIGNY
DURAND        13,RUE DE MILANPARIS
MARTIN        XXX            KKKK
RACLIN        RUE XXX        PARIS

Appuyer sur ENTER ?
```

GESTION DE FICHIER AUTOMATIQUE

Avec ce programme, la description des rubriques (NOM,LONGUEUR,TYPE) est faite par l'utilisateur de façon conversationnelle. Cette description est sauvegardée dans le fichier.

Une table FICH$(,) à 2 dimensions contient dans chaque ligne un "enregistrement".

NRUBS()	TYP$()	LG()	NOM	PR	RUE	VILLE	CPST
NOM	C	12	DUPONT	JEAN	RUE XXXX	MONTIGNY	78180
PR	C	10	MARTIN	DANIEL			
RUE	C	15	BALU	THIERRY			
VILLE	C	15					
CPST	C	5					

```
Nouveau fichier (O/N)? O

Nombre maxi de fiches? 100

Nom rubrique ? (ou <ENTER>) NOM
Type(C=chaine/ N=numérique)?C
Longueur maxi ? 12
Nom rubrique ? (ou <ENTER>) PR
Type(C=chaine/ N=numérique)?C
Longueur maxi ? 10
Nom rubrique ? (ou <ENTER>) RUE
Type(C=chaine/ N=numérique)?C
Longueur maxi ? 15
Nom rubrique ? (ou <ENTER>) VILLE
Type(C=chaine/ N=numérique)?C
Longueur maxi ? 15
Nom rubrique ? (ou <ENTER>) CPST
Type(C=chaine/ N=numérique)?C
Longueur maxi ? 5
Nom rubrique ? (ou <ENTER>)
Break in 360
Ok
```

CRÉATION ET MODIFICATION :

La création et la modification d'enregistrements se font avec le mode "C". Pour une modification, l'ancienne valeur de chaque rubrique est affichée. Le programme attend alors la nouvelle valeur. Si l'utilisateur appuie sur <ENTER> sans entrer de valeur, la rubrique n'est pas modifiée.

```
 LF:   liste du fichier
 LFA:  liste du fichier
 LFT:  liste du fichier triee
 FIN:  fin de session(SAUVEGARDE)
 S:    suppression fiche

Mode ? C

NOM ( ou <ENTER>)DURAND

Nouvelle clé OK (O/N)? O

R:zone arrière

DURAND         NOM     ?
               PR      ?JEAN
               RUE     ?11,RUE NOBEL
               VILLE   ?MONTGNY
               CPST    ?78180

NOM ( ou <ENTER>)
```

TRI / SÉLECTION :

Le mode "LFT" permet de sélectionner et de trier des fiches automatiquement.

```
Mode ? LFT
Position tri(nom rubrique)? NOM
Cle selection? (ou <ENTER>)
Nom rubrique a editer (ou <ENTER> ) NOM
Nom rubrique a editer (ou <ENTER> ) PR
Nom rubrique a editer (ou <ENTER> ) VILLE
Nom rubrique a editer (ou <ENTER> )

JE TRIE POUR VOUS
```

→

```
Liste triee des fiches

!BALU          !THIERRY     !ST CLOUD
!DUPOND        !JEAN        !MONTIGNY
!RACLIN        !JEAN        !PARIS

Appuyer sur <ENTER> ?
```

Vous pouvez ajouter des programmes spécifiques. Par exemple, un programme d'édition. Les rubriques sont connues sous le nom de FICH$(F,1), FICH$(F,2), etc.

```
3000 FOR F=1 TO NFICH
3010  PRINT FICH$(F,1);TAB(20)
3020  PRINT FICH$(F,2)
3040 NEXT F
```

Si vous n'effectuez qu'une sauvegarde, faire :

```
250 IF M$="FIN" THEN GOSUB 790:END
```

```
10 '------------------------------ GESTION DE FICHIER
20 '
30 '       Les noms des rubriques  sont definis de facon
40 '       conversationnelle et sauvegardes avec le fichier.
50 '       Une table FICH$(,) a 2 dimensions contient dans
60 '       chaque  ligne un 'enregistrement'.
70 '
80 '       MFICH:    nombre maximum de fiches
90 '       NFICH:    nombre de fiches
100 '      NRUB$():  noms des  rubriques
110 '
120 NFICH=0                          ' nombre de fiches
130 MODE 2:PRINT "CLAVIER MAJUSCULE"
140 INPUT "Nouveau fichier (O/N) ";R$:IF R$="O" OR R$="o" THEN G
OSUB 320:GOTO 160
150 GOSUB 930
160 CLS:PRINT "MODES:":PRINT
170 PRINT TAB(3);"C:      CREATION/MODIFICATION"
180 PRINT TAB(3);"LF:     LISTE DU FICHIER"
190 PRINT TAB(3);"FIN:    FIN DE SESSION(SAUVEGARDE)"
200 PRINT TAB(3);"LFA:    LISTE DU FICHIER"
210 PRINT TAB(3);"LFT:    LISTE DU FICHIER TRIEE"
220 PRINT TAB(3);"S:      SUPPRESSION FICHE"
230 PRINT:INPUT "MODE ";M$:M$=UPPER$(M$)
240 IF M$="C" THEN GOSUB 480
250 IF M$="FIN" THEN GOSUB 790:END
260 IF M$="LF" THEN GOSUB 1110
270 IF M$="LFA" THEN GOSUB 1310
280 IF M$="LFT" THEN GOSUB 1380
290 IF M$="S" THEN GOSUB 2130
300 GOTO 160
310 '========================== description des rubriques
320 PRINT
330 INPUT "Nombre maxi de fiches ";MFICH
340 PRINT
350 FOR K=1 TO 10                         ' 10 rubriques maxi
360  LINE INPUT "Nom rubrique  (ou <ENTER>) ";X$
370  IF X$="" THEN NRUB=K-1:DIM FICH$(MFICH,NRUB),CTRI$(MFICH),I
X%(MFICH):RETURN
380  NRUB$(K)=X$
390  LINE INPUT "Type (C=chaine/ N=numerique ) ? ",X$
```

→

```
400  IF X$<>"C" AND X$<>"N" THEN 390
410  TYP$(K)=X$
420  INPUT "Longueur maxi ";X
430  LG(K)=X
440 NEXT K
450 PRINT "TROP DE RUBRIQUES":STOP
460 '================================= CREATION/MODIFICATION
470 ' En mode modification,entrer seulement les premieres lettre
s de la cle
480 PRINT
490 PRINT NRUB$(1);
500 LINE INPUT " (ou <ENTER> ) ";CLE$:IF CLE$="" THEN RETURN
510 '                    pas d'espace entre "" de IF CLE$=""
520 LG=LEN(CLE$)
530 IF NFICH=0 THEN 620
540 FOR RANG=1 TO NFICH
550  IF CLE$=LEFT$(FICH$(RANG,1),LG) THEN 680  ' nom existe?
560 NEXT RANG
610 '-------------- nouvelle cle
620 PRINT:INPUT "Nouvelle cle OK (O/N) ";R$
630 IF R$<>"O" AND R$<>"o"  THEN 480
640 NFICH=NFICH+1
650 RANG=NFICH
660 FICH$(RANG,1)=CLE$
670 '----------------------- entree/modif zones
680 PRINT
690 PRINT "R:ZONE ARRIERE":PRINT
700 FOR R=2 TO NRUB                         ' ou FOR R=1 TO NRUB
710  PRINT FICH$(RANG,R);TAB(15);              ' ancienne valeur
720  PRINT NRUB$(R);TAB(22);                   ' nom de zone
730  LINE INPUT "?",X$
740  IF X$="R" THEN IF R>1 THEN R=R-1:GOTO 710
750  IF X$<>"" THEN FICH$(RANG,R)=X$
760 NEXT R
770 GOTO 480
780 '========================================== sauvegarde
790 OPENOUT "FICH"
800 PRINT #9,MFICH:PRINT #9,NFICH:PRINT #9,NRUB
810 FOR R=1 TO NRUB
820  PRINT #9,NRUB$(R):PRINT #9,TYP$(R):PRINT #9,LG(R)
830 NEXT R
840 '
850 FOR F=1 TO NFICH
860  FOR R=1 TO NRUB
870   PRINT #9,FICH$(F,R)
880 NEXT R
890 NEXT F
900 CLOSEOUT
910 RETURN
920 '==================================== lecture fichier
930 OPENIN "FICH"
940 '
950 INPUT #9,MFICH,NFICH,NRUB
960 DIM FICH$(MFICH,NRUB),CTRI$(MFICH),IX%(MFICH)
970 FOR R=1 TO NRUB
980  INPUT #9,NRUB$(R),TYP$(R),LG(R)
990 NEXT R
1000 FOR F=1 TO NFICH
1010  FOR R=1 TO NRUB
1020  LINE INPUT #9,FICH$(F,R)
```

→

```
1030  NEXT R
1040 NEXT F
1050 CLOSEIN
1060 PRINT:PRINT NFICH;"FICHES"
1070 FOR TP=1 TO 2000:NEXT TP
1080 GOSUB 2070
1090 RETURN
1100 '============================ liste du fichier directe
1110 CLS
1120 PRINT "LISTE DU FICHIER":PRINT
1130 '
1140 FOR F=1 TO NFICH
1150  IF F MOD 20=0 THEN INPUT "Appuyer sur ENTER ";X$
1160  PRINT FICH$(F,1);TAB(15);          ' zone 1
1170  PRINT FICH$(F,2)                   ' zone 2
1180 NEXT F
1190 PRINT:INPUT "APPUYER SUR <ENTER> ";X$
1200 RETURN
1300 '========================= liste des fiches automatique
1310 ID(1)=1:ID(2)=3            ' numeros des zones (a adapter)
1320 '
1330 CLS:PRINT "LISTE DES FICHES":PRINT
1340 FOR F=1 TO NFICH:IX%(F)=F:NEXT F
1350 NC=NFICH:GOSUB 1650
1360 RETURN
1370 '============================ tri/selection des fiches
1375 ' Pour cles commencant par 'BA' repondre 'BA'
1376 ' a la question 'Cle selection? '
1380 INPUT "Position tri (nom rubrique) ";X$:GOSUB 1490:IF R>NRU
B THEN 1380
1390 PTRI=R
1400 LINE INPUT "Cle selection? (ENTER pour toutes) ",CLE$
1410 FOR J=1 TO NRUB+1
1420  ID(J)=0
1430  LINE INPUT "Nom Rubrique a editer (ou ENTER ";X$:IF X$=""
THEN 1540
1440  GOSUB 1490:IF R>NRUB THEN 1420
1450  ID(J)=R
1460 NEXT J
1470 GOTO 1540
1480 '--
1490 FOR R=1 TO NRUB
1500  IF X$=NRUB$(R) THEN RETURN
1510 NEXT R
1520 RETURN
1530 '-----
1540 NC=0                             ' nc: nombre de cles
1550 LG=LEN(CLE$)
1560 FOR F=1 TO NFICH
1570  IF CLE$<>LEFT$(FICH$(F,PTRI),LG) THEN 1590
1580  NC=NC+1:CTRI$(NC)=FICH$(F,PTRI):IX%(NC)=F
1590 NEXT F
1600 GOSUB 1920        ' appel tri
1610 PRINT:PRINT "LISTE TRIEE DES FICHES ":PRINT
1620 GOSUB 1650
1630 RETURN
1640 '------------------ spgm edition ecran
1650 FOR W=1 TO 10
1660  IF ID(W)=0 THEN 1700
```

```
1670  PS(W)=PS(W-1)+LG(ID(W))+2
1680 NEXT W
1690 '--
1700 FOR F=1 TO NC
1710  F1=IX%(F)
1720  PRINT "!";
1730  FOR R=1 TO NRUB
1740   IF ID(R)=0 THEN 1790
1750   PRINT FICH$(F1,ID(R));
1760   PRINT TAB(PS(R));
1770   PRINT "!";
1780  NEXT R
1790 PRINT
1800 IF F MOD 20=0 THEN INPUT "APPUYER SUR ENTER";X$
1810 NEXT F
1820 PRINT:INPUT "APPPUYER SUR <ENTER> ";X$
1830 RETURN
1910 '------------------------------- tri SHELL-METZNER
1920 ECART=NC
1930 PRINT:PRINT "JE TRIE POUR VOUS ":PRINT
1940 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
1950 J=1 :K=NC-ECART
1960 L=J
1970 M=L+ECART
1980 IF CTRI$(L)<=CTRI$(M) THEN 2040
1990 X$=CTRI$(L):CTRI$(L)=CTRI$(M):CTRI$(M)=X$
2000 X=IX%(L):IX%(L)=IX%(M):IX%(M)=X
2010 L=L-ECART:IF L<1 THEN 2040
2020 GOTO 1970
2030 '
2040 J=J+1:IF J>K THEN 1940
2050 GOTO 1960
2060 '--------------- liste descripteur
2070 FOR R=1 TO NRUB
2080  PRINT NRUB$(R)
2090 NEXT R
2100 FOR TP=1 TO 2000:NEXT TP
2110 RETURN
2120 '================================== suppression fiche
2130 PRINT :PRINT NRUB$(1);:LINE INPUT "?",CLE$
2140 IF CLE$="" THEN RETURN
2150 FOR RANG=1 TO NFICH
2160  IF CLE$=FICH$(RANG,1) THEN  2200
2170 NEXT RANG
2180 PRINT:PRINT "N'existe pas":GOTO 2130
2190 '
2200 PRINT:INPUT "ANNULE OK(O/N) ";R$:IF R$<>"O" THEN 2130
2210 FOR J=RANG TO NFICH-1
2220  FOR R=1 TO NRUB
2230   FICH$(J,R)=FICH$(J+1,R)
2240  NEXT R
2250 NEXT J
2260 FOR R=1 TO NRUB:FICH$(NFICH,R)="":NEXT R
2270 NFICH=NFICH-1
2280 GOTO 2130
```

BIBLIOTHÈQUE

Une bibliothèque est gérée dans des tables en mémoire centrale. Ces tables sont sauvegardées sur cassette.

TITR$( )

MA VIE ET LA PSYCHANALYSE
PROGRAMMER EN ASSEMBLEUR

Titres

AUT$( )

FREUD
PINAUD

Auteur

C$(,)

PSYCHANALYSE	
ASSEMBLEUR	Z80

Deux mots-clés par ouvrage sont prévus. L'édition des listes triées par titre et par mot-clé se fait selon le principe présenté dans le programme de gestion d'adresses.

On remarquera l'instruction :

```
340 IF TITR$=LEFT$(TITR$(RANG,L)) THEN 460
```

En "mode modification", elle permet d'entrer seulement les premières lettres du titre.

```
  C:        Création modification
  LTITRE:   Liste par titre
  LCLE:     Liste par mot-clé
  FIN:      Sauvegarde
  S:        Suppression fiche

Mode ? C


Titre? PROGRAMMER EN ASSEMBLEUR

Nouveau titre (O/N) ? O

PROGRAMMER EN ASSEMBLEUR

           Auteur ? PINAUD
           Mot clé ? ASSEMBLEUR
           Mot clé ? Z80

Titre?
Break in 280
Ok
Liste triée par titre

BASIC A Z            DUPONT
BASIC ET SES FICHIER DUPONT
BASIC POUR TOUS      DUPONT
PROGRAMMER EN ASSEMB PINAUD

Appuyer sur <ENTER> ?
```

```
ASSEMBLEUR

   PROGRAMMER EN ASSEMB PINAUD

BASIC

   BASIC ET SES FICHIER DUPONT
   BASIC A Z            DUPONT
   BASIC POUR TOUS      DUPONT

FICHIER

   BASIC ET SES FICHIER DUPONT

Z80

   PROGRAMMER EN ASSEMB PINAUD
```

Si vous n'effectuez qu'une sauvegarde, faire :

```
200 IF M$="FIN" THEN GOSUB 980:END
```

```
10 '------------------------------ BIBLIOTHEQUE
20 MODE 2:PRINT "CLAVIER MAJUSCULE"
30 DIM TITR$(100),AUT$(100),C$(100,2)
40 DIM CLE$(200),IX%(200)
50 NFICH=0                              nombre de fiches
60 INPUT "NOUVEAU FICHIER (O/N) ";R$
70 IF R$="O" OR R$="o" THEN 100
80 GOSUB 1110
90 '
100 CLS
110 CLS:PRINT "MODES:":PRINT
120 PRINT TAB(3) "C:     CREATION/MODIFICATION"
130 PRINT TAB(3) "LTITRE:LISTE PAR TITRE"
140 PRINT TAB(3) "LCLE:  LISTE PAR MOT-CLE"
150 PRINT TAB(3) "FIN:   SAUVEGARDE"
160 PRINT TAB(3) "S:     SUPPRESSION FICHE"
170 PRINT:INPUT "MODE ";M$:M$=UPPER$(M$)
180 IF M$="C" THEN GOSUB 270
190 IF M$="LTITRE" THEN GOSUB 660
200 IF M$="FIN" THEN GOSUB 980:END
210 IF M$="LCLE" THEN GOSUB 1310
220 IF M$="S" THEN GOSUB 1550
230 GOTO 110
240 '=============================== creation/modification
250 ' en mode modification,entrer les premieres lettres du titre
260 '
270 PRINT
280 PRINT:LINE INPUT "TITRE? (OU ENTER) ";TITR$
290 IF TITR$="" THEN RETURN
300 '
310 LG=LEN(TITR$)
320 IF NFICH=0 THEN 370
330 FOR RANG=1 TO NFICH
340  IF TITR$=LEFT$(TITR$(RANG),LG) THEN 460
350 NEXT RANG
360 '------------------------- nouveau titre
```

→

```
370 PRINT:INPUT "NOUVEAU TITRE (O/N) ";R$
380 IF R$<>"O" AND R$<>"o" THEN 280
390 NFICH=NFICH+1
400 RANG=NFICH
410 '
420 TITR$(RANG)=TITR$
430 '--
440 ' pour modification,appuyer sur <ENTER> si zone sans changem
ent
450 '
460 PRINT :PRINT TITR$(RANG):PRINT
470 PRINT AUT$(RANG);TAB(15);                   ancienne valeur
480 INPUT "AUTEUR ";AUT$:IF AUT$<>"" THEN AUT$(RANG)=AUT$
490 '
500 FOR C=1 TO 2
510  PRINT C$(RANG,C);TAB(15);                  ancienne valeur
520  INPUT "MOT CLE ";C$
530  IF C$<>"" THEN C$(RANG,C)=C$
540 NEXT C
550 GOTO 280
630 '============================ liste triee par titre
640 '  Pour liste des ouvrages commencant par 'BA' ,repondre 'BA
' a 'Cle?'
650 '
660 PRINT:INPUT "CLE (ENTER pour tous les titres) ";CLE$
670 LG=LEN(CLE$)
680 NC=0                             ' nc:nombre de cles
690 FOR LV=1 TO NFICH
700  IF CLE$<>LEFT$(TITR$(LV),LG) THEN 720
710  NC=NC+1:CLE$(NC)=TITR$(LV):IX%(NC)=LV
715  IF F MOD 20=0 THEN INPUT "APPUYER SUR ENTER";X$
720 NEXT LV
730 GOSUB 860                          ' appel tri
740 '------------------ edition
750 CLS:PRINT "LISTE TRIEE PAR TITRE ":PRINT
760 IF NC=0 THEN RETURN
770 FOR LV=1 TO NC
780  X=IX%(LV)
790  PRINT LEFT$(TITR$(X),20);TAB(22);
800  PRINT AUT$(X)
810  IF LV MOD 20=0 THEN INPUT X$
820 NEXT LV
830 PRINT:INPUT "APPUYER SUR <ENTER> ";X$
840 RETURN
850 '--------------------------- tri shell
860 ECART=NC
870 '
880 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
890 IV=0
900 FOR K=1 TO NC-ECART
910  J=K+ECART
920  IF CLE$(J)>=CLE$(K) THEN 950
930  X$=CLE$(K):CLE$(K)=CLE$(J):CLE$(J)=X$:IV=1
940  X=IX%(K):IX%(K)=IX%(J):IX%(J)=X
950 NEXT K
960 IF IV=1 THEN 890 ELSE 880
970 '================================ sauvegarde cassette
980 OPENOUT "BIB"
990 IF NFICH=0 THEN RETURN
1000 PRINT #9,NFICH
```

→

```
1010 FOR F=1 TO NFICH
1020 PRINT #9,TITR$(F)
1030  PRINT #9,AUT$(F)
1040  FOR C=1 TO 2
1050   PRINT #9,C$(F,C)
1060  NEXT C
1070 NEXT F
1080 CLOSEOUT
1090 RETURN
1100 '======================================= lecture cassette
1110 OPENIN "BIB"
1120 INPUT #9,NFICH
1130 FOR F=1 TO NFICH
1140  INPUT #9,TITR$(F)
1150  INPUT #9,AUT$(F)
1160  FOR C=1 TO 2
1170   INPUT #9,C$(F,C)
1180  NEXT C
1190 NEXT F
1200 CLOSEIN
1210 PRINT:PRINT NFICH;"TITRES":PRINT
1220 FOR TP=1 TO 2000:NEXT TP
1230 RETURN
1280 '================================ liste par mot cle
1290 ' Pour tous les mot-cle,appuyer sur <ENTER> pour la questio
n 'Cle?'
1300 '
1310 INPUT "CLE (ENTER pour toutes ) ";CLE$
1320 LG=LEN(CLE$)
1330 NC=0                                     ' nombre de cles
1340 FOR F=1 TO NFICH
1350  FOR C=1 TO 2
1360   IF C$(F,C)="" THEN 1390
1370   IF CLE$<>LEFT$(C$(F,C),LG) THEN 1390
1380   NC=NC+1:CLE$(NC)=C$(F,C):IX%(NC)=F
1390  NEXT C
1400 NEXT F
1410 '
1420 GOSUB 860                                ' appel tri
1430 '---------------------- edition
1440 CLS:PRINT "LISTE TRIEE PAR MOT-CLE":PRINT
1450 FOR F=1 TO NC
1460  X=IX%(F)
1470  IF CLE$(F-1)<>CLE$(F) THEN PRINT:PRINT CLE$(F):PRINT
1480  PRINT TAB(4);LEFT$(TITR$(X),20);TAB(25);
1490  PRINT AUT$(X)
1500  IF F MOD 8=0 THEN INPUT "APPUYER SUR ENTER ";X$
1510 NEXT F
1520 PRINT:INPUT "APPUYER SUR <ENTER>";X$
1530 RETURN
1540 '================================== suppression titre
1550 PRINT:INPUT "TITRE ";TITR$:IF TITR$="" THEN RETURN
1560 LG=LEN(TITR$)
1570 FOR RANG=1 TO NFICH
1580  IF TITR$=LEFT$(TITR$(RANG),LG) THEN 1620
1590 NEXT RANG
1600 PRINT:PRINT "N'existe pas":PRINT:GOTO 1550
1610 '
1620 PRINT TITR$(RANG):PRINT
1630 INPUT "SUPPRESSION OK (O/N) ) ";R$
```

→

```
1640 IF R$<>"O" THEN 1550
1650 FOR J=RANG TO NFICH-1
1660  TITR$(J)=TITR$(J+1)
1670  AUT$(J)=AUT$(J+1)
1680  FOR C=1 TO 2
1690   C$(J,C)=C$(J+1,C)
1700  NEXT C
1710 NEXT J
1720 TITR$(NFICH)="":AUT$(NFICH)=""
1730 FOR C=1 TO 2:C$(NFICH,C)="":NEXT C
1740 NFICH=NFICH-1
1750 GOTO 1550
```

LE CPC 664 | 14

Le BASIC du CPC 664 est très proche de celui du CPC 464. Il dispose de quelques instructions supplémentaires. Pour être en mesure de sauvegarder vos programmes BASIC, vous devez lire l'introduction de cette partie.

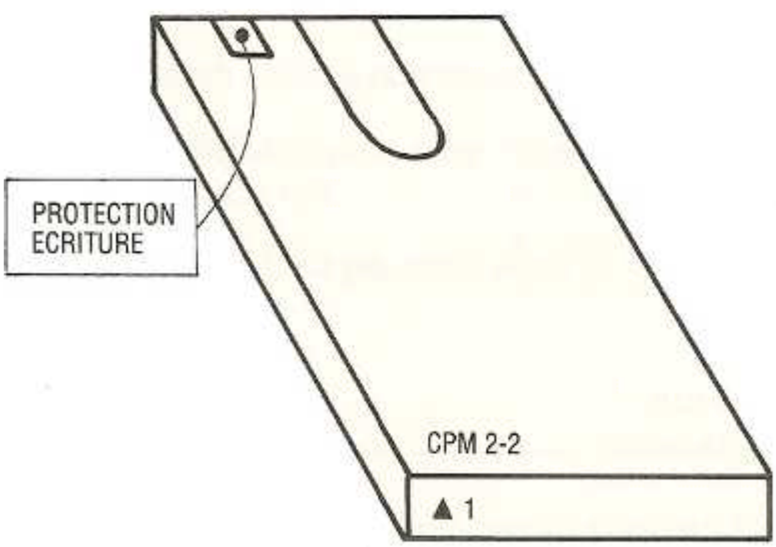
INTRODUCTION

Lorsque vous mettez le CPC 664 sous tension, le message ci-dessous apparaît :

```
Amstrad 64K Microcomputer  (v2)

©1984 Amstrad Consumer Electronics plc
         and Locomotive Software Ltd.

BASIC 1.1
```



CAT

Vous êtes sous BASIC. Placez la disquette CP/M dans le lecteur de disquette. La commande "CAT" vous permet d'afficher la liste des fichiers de la disquette :

```
Ready
CAT

Drive A: user  0

AMSDOS  .COM  1K   DDT     .COM   5K   FILECOPY.COM   3K   STAT    .COM  6K
ASM     .COM  8K   DISC    .BAS  20K   FORMAT  .COM   3K   SUBMIT  .COM  2K
BOOTGEN .COM  2K   DISCCHK .COM   3K   LETTRE  .TXT   1K   SYSGEN  .COM  2K
CHKDISC .COM  3K   DISCCOPY.COM   3K   LOAD    .COM   2K   X       .TXT  1K
CLOAD   .COM  2K   DUMP    .ASM   5K   MOVCPM  .COM  10K   XSUB    .COM  1K
CODE    .$$$  0K   DUMP    .COM   1K   PIP     .COM   8K   XX      .TXT  1K
COPIE   .     2K   ED      .COM   7K   RITDEMO .BIN  12K   XY      .TXT  1K
COPYDISC.COM  3K   EX1     .BAS   2K   ROINTIME.DEM  26K
CSAVE   .COM  2K   EX2     .BAS   1K   SETUP   .COM   8K

 12K free
```

Pour sauvegarder vos programmes, vous devez d'abord initialiser une disquette vierge. Pour cela, accédez à CP/M en frappant "ICPM" :

```
ICPM
>CP/M version 2.2
```

Ensuite, frappez "FORMAT" (avec la disquette CP/M). Enfin, placez votre disquette vierge et appuyez sur une touche quelconque.

```
A>FORMAT

FORMAT V2.0

Please insert disc to be formatted into drive A
then press any key:_
Formatting started
Formatting track   0
Formatting track   1
Formatting track   2
Formatting track   3
```

Remarque : Assurez-vous que la protection écriture de la disquette n'est pas activée.

Lorsque la disquette est "formatée", vous pouvez sauvegarder vos programmes avec la commande SAVE.

Pour revenir au BASIC, placez dans disquette CP/M puis frappez "AMSDOS".

SAVE "nom-programme"
LOAD "nom-programme"

La commande SAVE sauvegarde sur disquette le programme en mémoire centrale.

La commande LOAD permet de le transférer de la disquette vers la mémoire centrale.

```
10 PRINT "AMSTRAD"
20 PRINT "CPC664"

SAVE "ESSAI
Ready

NEW
Ready

LOAD "ESSAI
Ready
LIST
10 PRINT "AMSTRAD"
20 PRINT "CPC664"
```

Le programme est sauvegardé avec le type ".BAS" et s'appelle en réalité "ESSAI.BAS". Pour l'effacer, vous devrez spécifier le nom suivi du type.

Lorsqu'un programme est sauvegardé une seconde fois, l'ancienne version est sauvegardée avec le type ".BAK". Pour l'effacer, frappez :

|ERA, "ESSAI.BAK"

Comme sur cassette, un programme peut être sauvegardé en ASCII par :

SAVE "non-programme", A

(cf. chapitre commandes).

Nous vous conseillons de dupliquer la disquette CP/M sur une autre disquette (cf. DISCCOPY plus loin).

COMMANDES ACCESSIBLES SOUS BASIC 6128 et 664

- **CAT**
- **|DIR**
- **|ERA**
- **|TAPE.IN**
- **|TAPE.OUT**
- **|TAPE**
- **|DISC.IN**
- **|DISC.OUT**
- **|DISC**
- **|A**
- **|B**
- **|DRIVE,"A"**
- **|DRIVE,"B"**
- **|REN**
- **|USER**
- **|CPM**

Cf. Chapitre CPC 6128.

COMMANDES ACCESSIBLES SOUS CP/M 2.2

- **DIR**
- **DISCCOPY**
- **FORMAT**
- **FILECOPY**
- **COPYDISC**
- **ERA**
- **STAT**
- **PIP**
- **TYPE**
- **REN**
- **CLOAD**
- **CSAVE**
- **SET UP**
- **AMSDOS**

L'accès à CP/M se fait en frappant "|CPM". Un "type" est attribué à la plupart des fichiers. Par exemple, lorsqu'un programme BASIC est sauvegardé par SAVE "PROG", le type ".BAS" lui est affecté ; le nom du fichier pour CP/M est "PROG.BAS".

Ceci permet de regrouper les fichiers par famille. Les types les plus courants sont :

.BAS	BASIC
.BIN	BINAIRE
.BAK	SAUVEGARDE
.COM	COMMANDE

Pour référencer une famille de fichiers, il suffit de spécifier le signe "*" à la place du nom.

Les commandes DIR, ERA, REN, TYPE sont exécutables sans la disquette CP/M. Les autres commandes doivent être présentes sur la disquette lorsqu'elles sont appelées. Vous pouvez recopier certaines commandes de la disquette CP/M sur votre disquette de programmes. L'exécution d'une commande est annulée en frappant CTRL/C.

DIR

Affiche la liste des fichiers d'une disquette.

`|DIR`	affiche la liste de tous les fichiers de la disquette du lecteur A:
`|DIR *.BAS`	affiche la liste des fichiers du type ".BAS".
`|DIR B:`	affiche la liste des fichiers de la disquette du lecteur B:

```
A>DIR
A: MOVCPM   COM : PIP      COM : SUBMIT   COM : XSUB     COM
A: ED       COM : ASM      COM : DDT      COM : LOAD     COM
A: STAT     COM : DUMP     COM : DUMP     ASM : AMSDOS   COM
A: FILECOPY COM : SYSGEN   COM : BOOTGEN  COM : COPYDISC COM
A: CHKDISC  COM : DISCCOPY COM : DISCCHK  COM : SETUP    COM
A: FORMAT   COM : CSAVE    COM : CLOAD    COM : EX1      BAS
A: EX2      BAS : ROINTIME DEM : RITDEMO  BIN : DISC     BAS
A>
```

DISCCOPY

Copie une disquette sur une autre avec un seul lecteur de disquette. La disquette "destination" n'a pas à être formatée. Placer la disquette CP/M et frapper "DISCCOPY". Placer ensuite la disquette source à recopier et appuyer sur une touche.

```
A>DISCCOPY

DISCCOPY V2.0

Please insert source disc into drive A then press any key:_
Copying started
Reading track  0
Reading track  1
Reading track  2
Reading track  3
Reading track  4
Reading track  5
Reading track  6
Reading track  7
Please insert destination disc into drive A then press any key:_
Writing track  0
Writing track  1
```

La sauvegarde est faite par groupes de 8 pistes. Vous devez alternativement placer les disquettes "source" et "destination" dans le lecteur.

FORMAT

Permet de "formater" une disquette vierge. Placer la disquette CP/M contenant le fichier FORMAT.COM et frapper :

FORMAT

Placer ensuite la disquette vierge dans le lecteur et appuyer sur une touche quelconque. La disquette formatée est alors utilisable par AMSTRAD.

FILECOPY nom-fichier

Copie un fichier ou un groupe de fichiers d'une disquette "source" vers une disquette "destination". La disquette destination doit être formatée.

Placer la disquette CP/M dans le lecteur et frapper FILECOPY. Placer ensuite la disquette source contenant les fichiers à recopier.

```
A>FILECOPY *.BAS

FILECOPY V2.1

Please insert SOURCE disc into drive A then press any key:_

Ambiguous file name:  Confirm individual files (Y/N) ? Y

EX1     .BAS  Copy (Y/N) ? Y
EX2     .BAS  Copy (Y/N) ? N
DISC    .BAS  Copy (Y/N) ? Y
XX      .BAS  Copy (Y/N) ? N

Copying started. . . .

Please insert DESTINATION disc into drive A then press any key:_
```

COPYDISC

Copie une disquette du lecteur A: vers le lecteur B:

ERA nom-fichier

Efface un fichier ou un groupe de fichiers.

`ERA PROG.BAS`	efface le fichier PROG.BAS.
`ERA *.BAS`	efface tous les fichiers du type .BAS.
`ERA *.*`	efface tous les fichiers.

STAT

Fournit les caractéristiques des fichiers.

`STAT` donne la place libre sur la disquette.

```
A>STAT
A: R/O, Space: 54k
```

`STAT *.*` donne les caractéristiques de tous les fichiers.

```
A>STAT *.*

 Recs  Bytes  Ext Acc
    2     1k    1 R/W A:AMSDOS.COM
   64     8k    1 R/W A:ASM.COM
   10     2k    1 R/W A:BOOTGEN.COM
```

`STAT *.BAS` donne la liste pour les fichiers du type .BAS.

```
A>STAT *.BAS

 Recs  Bytes  Ext Acc
  158    20k    2 R/W A:DISC.BAS
   11     2k    1 R/W A:EX1.BAS
    3     1k    1 R/W A:EX2.BAS
    4     1k    1 R/W A:XX.BAS
Bytes Remaining On A: 54k
```

`STAT *.COM $R/O` les fichiers du type "COM" ne peuvent être supprimés.

```
A>STAT *.COM $R/O

MOVCPM.COM set to R/O
PIP.COM set to R/O
SUBMIT.COM set to R/O
XSUB.COM set to R/O
```

`STAT *.COM $R/W` les fichiers du type "COM" peuvent être supprimés.

```
A>STAT *.COM $R/W

MOVCPM.COM set to R/W
PIP.COM set to R/W
SUBMIT.COM set to R/W
```

`STAT *.COM $SYS` les fichiers du type "COM" n'apparaissent plus lors d'une commande DIR.

```
A>STAT  *.COM $SYS

MOVCPM.COM set to SYS
PIP.COM set to SYS
SUBMIT.COM set to SYS
XSUB.COM set to SYS
```

`STAT  *.COM $DIR` les fichiers du type "COM" apparaissent à nouveau lors d'une commande "DIR".

```
A>STAT *.COM $DIR

MOVCPM.COM set to DIR
PIP.COM set to DIR
SUBMIT.COM set to DIR
XSUB.COM set to DIR
```

PIP destination=source

Permet de copier un ou plusieurs fichiers.

`PIP COPIE=EX1.BAS` copie le fichier EX1.BAS sous le nom de COPIE.

Les périphériques peuvent être considérés comme des fichiers :

LST:	IMPRIMANTE
CON:	CLAVIER ET ECRAN
RDR:	INTERFACE SERIE ENTREE
PUN:	INTERFACE SERIE SORTIE

Ci-dessous, le fichier LETTRE.TXT est affiché sur l'imprimante.

```
A>PIP LST:=LETTRE.TXT

Cher Monsieur
Veuillez trouver ci joint.....
```

Ci-dessous, nous frappons une lettre au clavier. Les sauts de ligne sont obtenus en frappant CTRL/J. Pour terminer, frapper CTRL/Z.

```
A>PIP LETTRE.TXT=CON:

CHER MONSIEUR

VEUILLEZ TROUVER CI JOINT ....
^O

A>TYPE LETTRE.TXT

CHER MONSIEUR

VEUILLEZ TROUVER CI JOINT ....
```

TYPE nom-fichier

Affiche à l'écran le fichier spécifié.

TYPE LETTRE. TXT

Un programme BASIC ne peut être affiché que s'il a été sauvegardé en ASCII.

REN nouveau nom=ancien nom

Change le nom d'un fichier.

REN Y.BAS=X.BAS

Le fichier X.BAS devient Y.BAS.

CLOAD "nom-fichier-cassette" nom-fichier-disquette

Copie un fichier cassette sur disquette. Le fichier doit être du type ASCII. Un programme doit avoir été sauvegardé par "SAVE "nom",A".

```
A>CLOAD "" XX.TXT

CLOAD V2.0


CLOAD V2.0 finished
```

Un programme BASIC sur cassette ou un fichier binaire peuvent être copiés sous BASIC en utilisant |TAPE.IN et LOAD"".

CSAVE nom-fichier-disquette "nom-fichier-cassette"

Copie un fichier ASCII disquette sur cassette.

SET UP

Permet d'afficher et de changer les paramètres définis pour le système AMSTRAD.

Ci-dessous, nous avons défini une commande (STAT) qui est exécutée à chaque fois que CP/M est activé.

" ↑ M" définit le caractère retour-chariot.

```
A>SETUP

SETUP V2.0
** Initial command buffer empty

Is this correct (Y/N):_N

Enter new initial command buffer:_STAT↑M
Initial command buffer:

STAT^M

Is this correct (Y/N):_YSign-on string:

Is this correct (Y/N):_Y
Do you want to update your system disc (Y/N):_Y
```

AMSDOS

Cette commande provoque le retour au BASIC.

CARACTERES DE CONTROLE CP/M

CTRL/C	interrompt l'exécution d'une commande.
CTRL/J	saut de ligne.
CTRL/M	retour en début de ligne.
CTRL/P	provoque l'affichage sur imprimante.
CTRL/S	suspend l'exécution d'une commande qui se poursuit en appuyant sur une touche.
CTRL/Z	définit la fin d'un fichier.

LE CPC 6128 | 15

Le BASIC du CPC 6128, identique à celui du CPC 664, est très proche de celui du CPC 464. Il comporte quelques instructions supplémentaires.

Pour être en mesure de sauvegarder vos programmes BASIC, vous devez lire l'introduction de ce chapitre.

INTRODUCTION

Lorsque vous mettez le CPC 6128 sous tension, le message suivant apparaît :

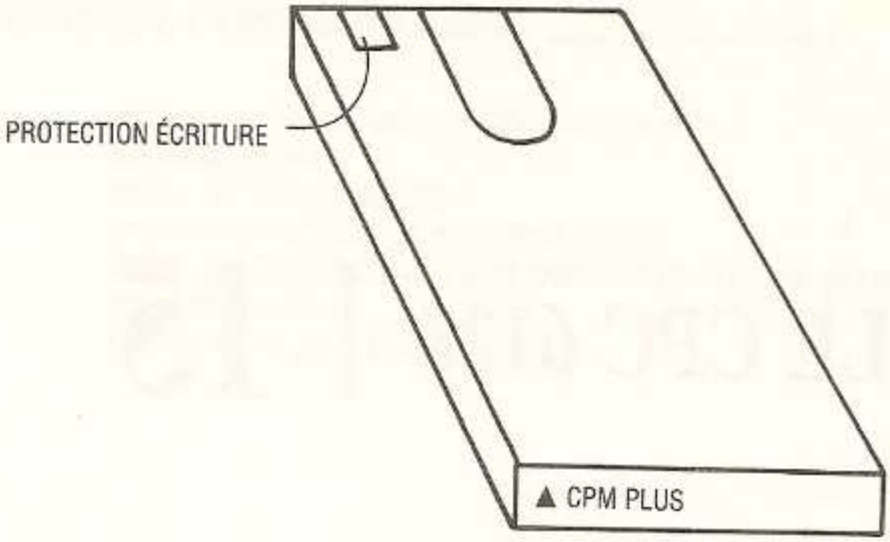
```
 Amstrad 128K Microcomputer  (v3)

 ©1985 Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

 BASIC 1.1

Ready
```

La disposition de certaines touches du clavier diffère de celle du CPC 464. La touche ENTER du CPC 464 devient RETURN sur CPC 6128. La touche ENTER du CPC 6128 est équivalente à celle du pavé numérique sur le CPC 464.



CAT

Vous êtes sous BASIC. Placez la disquette CP/M Plus dans le lecteur de disquette. La commande "CAT" vous permet d'afficher la liste des fichiers de la disquette.

```
CAT

Drive A: user  0

AMSDOS  .COM   1K  DISCKIT3.COM   6K  PALETTE .COM   1K  SETDEF  .COM   4K
BANKMAN .BAS   1K  ED      .COM  10K  PIP     .COM   9K  SETKEYS .COM   2K
BANKMAN .BIN   2K  ERASE   .COM   4K  PROFILE .ENG   1K  SETLST  .COM   2K
C10CPM3 .EMS  25K  GET     .COM   7K  PUT     .COM   7K  SETSIO  .COM   2K
DATE    .COM   3K  KEYS    .CCP   1K  RENAME  .COM   3K  SHOW    .COM   9K
DEVICE  .COM   8K  KEYS    .WP    1K  SET     .COM  11K  SUBMIT  .COM   6K
DIR     .COM  15K  LANGUAGE.COM   1K  SET24X80.COM   1K  TYPE    .COM   3K

 23K free
```

Pour sauvegarder vos programmes, vous devez d'abord "formater" une disquette vierge. Pour cela, accédez à CPM Plus en frappant "|CPM" (le signe | s'obtient avec SHIFT @).

```
|CPM ↵

CP/M Plus Amstrad Consumer Electronics plc
V 1.0 ,61 TPA , 1 disc drive
```

Ensuite, frappez "DISCKIT3" (avec la disquette CP/M Plus). Un premier menu apparaît, placez votre disquette vierge à formater dans le lecteur.

A>DISCKIT3 ↵

copy	7
format	4
verify	1
exit	0

- Assurez-vous que la protection écriture de la disquette n'est pas activée.
- Appuyez sur la touche "f4" qui vous permet de choisir le mode "format".
- Un second menu apparaît :

system format	9
data format	6
vendor format	3
exit menu	.

- Appuyez sur la touche "f6".
- Le message ci-dessous apparaît :

```
Y    format as data

     Any key to exit to menu
```

- Appuyez sur "Y" pour confirmer (Y=yes).
- Les numéros des pistes formatées défilent en haut à gauche de l'écran.
- Lorsque les 40 pistes de la disquette sont formatées, le message "REMOVE DISC" apparaît.
- Enlevez votre disquette et appuyez sur une touche quelconque pour revenir au menu principal. Appuyez alors sur "f0" pour revenir à CP/M Plus.

Vous pouvez désormais sauvegarder des programmes sur votre disquette. Pour revenir au BASIC, placez la disquette CP/M Plus et frappez "AMSDOS".

SAVE "nom-programme"
LOAD "nom-programme"

La commande "SAVE" sauvegarde sur disquette le programme en mémoire centrale.

La commande "LOAD" permet de le transférer de la disquette vers la mémoire centrale.

```
10 PRINT "AMSTRAD"
20 PRINT "CPC6128"

SAVE "ESSAI"

NEW

LOAD "ESSAI"

LIST

10 PRINT "AMSTRAD"
20 PRINT "CPC6128"
```

Le programme est sauvegardé avec le type " BAS" et s'appelle en réalité "ESSAI.BAS". Pour l'effacer, vous devrez spécifier le nom suivi du type.

Lorsqu'un programme est sauvegardé une seconde fois, l'ancienne version est sauvegardée avec le type ".BAK". Pour l'effacer, frapper :

|ERA, "ESSAI.BAK"

Comme sur cassette, un programme peut être sauvegardé en ASCII par :

SAVE "non-programme", A

(cf. chapitre commandes).

Nous vous conseillons de dupliquer la disquette CP/M sur une autre disquette (avec l'option COPY de DISCKIT3).

COMMANDES ACCESSIBLES SOUS BASIC 6128 et 664

- CAT
- |DIR
- |ERA
- |TAPE.IN
- |TAPE.OUT
- |TAPE
- |DISC.IN
- |DISC.OUT
- |DISC
- |A
- |B
- |DRIVE,"A"
- |DRIVE,"B"
- |REN
- |USER
- |CPM

Toutes ces commandes (sauf CAT) doivent être précédées du signe "|". Si vous disposez d'un second lecteur de disquette, celui-ci doit être spécifié par "B:" devant le nom du fichier.

Les commandes peuvent être incluses comme instructions dans un programme.

CAT

Donne la liste des fichiers d'une disquette.

|DIR

Donne la liste des fichiers d'une disquette.

`DIR`	donne la liste des fichiers de la disquette du lecteur A:
`DIR ,"*.BAS"`	donne la liste de tous les fichiers du type ".BAS".
`DIR ,"B:"`	donne la liste des fichiers de la disquette du lecteur "B:".

```
|DIR ,"*.BAS"

Drive A: user  0

EPSON   .BAS  DERR    .BAS  CURSOR  .BAS  SQUASH  .BAS  FDIS    .BAS
H3D     .BAS  COPYCHR$.BAS  HISTB   .BAS  MASK    .BAS  FILL    .BAS
TD      .BAS  GPAPER  .BAS  GPEN    .BAS  DEC     .BAS  LOCO    .BAS
HISTC   .BAS  IMMEUBLE.BAS  MAISON  .BAS  ETOILE  .BAS  TAG1    .BAS
```

|ERA, "nom fichier"

Efface le fichier ou le groupe de fichiers spécifiés. Le type doit être spécifié. Si vous avez sauvegardé un programme BASIC par SAVE "ESSAI", vous devez frapper :

```
|ERA ,"ESSAI.BAS"
```

`|ERA ,"PROG.BAS"` efface le programme "PROG".

`|ERA ,"*.BAS"` efface tous les fichiers du type ".BAS".

`|ERA ,"*.*"` efface tous les fichiers.

|TAPE.IN

Permet de lire un programme ou un fichier binaire d'une cassette. Frappez "|TAPE.IN" puis "LOAD "nom-programme"" :

```
Ready
|TAPE.IN
Ready
LOAD ""
Press PLAY then any key:
```

Après avoir sauvegardé le programme sur disquette, frappez "|DISC.IN" pour accéder au disque en lecture.

Pour accéder à un fichier séquentiel sur cassette par programme, inclure "|TAPE.IN" dans le programme.

```
10 |TAPE.IN
20 OPENIN "FICH"
```

|TAPE.OUT

Pour sauvegarder un programme ou la mémoire sur cassette, frappez "TAPE.OUT" puis "SAVE "non-programme"".

```
|TAPE.OUT
SAVE "PROG"
```

|TAPE

Est équivalent à "TAPE.IN" plus "|TAPE.OUT".

|DISC.IN

Annule "|TAPE.IN" et permet ainsi de référencer le lecteur de disquette en lecture.

|DISC.OUT

Annule "|TAPE.OUT".

|DISC

Est équivalent à "|DISC.IN" plus "|DISC.OUT".

|A

Définit le lecteur "A:" par défaut.

|B

Définit le lecteur "B:" par défaut.

|DRIVE ,"A"

Définit le lecteur "A:" par défaut.

|DRIVE ,"B:"

Définit le lecteur "B:" par défaut.

|REN ,"nouveau nom","ancien nom"

Change le nom d'un programme.

```
|REN ,"Y.BAS","X.BAS"
```

|USER ,numéro

Permet de définir plusieurs catalogues sur une disquette. Les catalogues sont repérés par des numéros allant de 0 à 15. Le numéro par défaut est égal à 0. Ci-dessous, nous listons les fichiers du catalogue numéro 2.

```
|USER ,2
Ready
|DIR

Drive A: user  2

CECRAN  .BAS

 52K free
```

|CPM

Provoque le passage sous CP/M.

UTILISATION DU SECOND BLOC MÉMOIRE

BASIC n'utilise que 64 k de la mémoire centrale. Les 64 k restants sont utilisés comme "banque de données" ou pour la sauvegarde de l'écran.

BANQUE DE DONNÉES

La "banque de données" est découpée en "enregistrements" auxquels on accède par un numéro. La longueur des enregistrements ne doit pas excéder 255 caractères.



| BANKOPEN ,longueur enregistrement

BANKOPEN définit la longueur des enregistrements.
|BANKOPEN, 50

| BANKWRITE ,@code retour,chaîne,numéro enregistrement

Ecrit la chaîne spécifiée au numéro d'enregistrement indiqué. Le code retour prend une valeur égale au numéro d'enregistrement si l'opération s'est bien déroulée.

– 1 indique que la taille des données dépasse 64 k.
– 2 indique une erreur de changement de bloc.

| BANKREAD ,@code retour,chaîne,numéro enregistrement

Lit l'enregistrement indiqué dans la chaîne spécifiée. Le code retour prend les mêmes valeurs que pour l'écriture.

Ci-dessous, nous écrivons trois noms dans la banque de données. Ensuite, nous recherchons le contenu des enregistrements.

Pour utiliser la banque de données, il est nécessaire de faire RUN "BANKMAN". Naturellement, le programme "BANKMAN.BAS" doit être présent sur la disquette. La variable code retour (R) doit être initialisée.

```
10 ' ESSAI BANKMAN
20 '
30 '   FAIRE RUN "BANKMAN" AVANT D'UTILISER CE PROGRAMME
40 '
50 MODE 2
60 '------------------------------ ECRITURE
70 |BANKOPEN ,50                               ' longueur=50
80 R%=0
90 A$="DUPONT Jean 044-22-63":N=1              ' enreg no 1
100 A$=LEFT$(A$+SPACE$(30),30)                 ' normalisation
110 |BANKWRITE ,@R%,A$,N
120 A$="MARTIN Daniel 333-44-56":N=2           ' enreg no 2
130 A$=LEFT$(A$+SPACE$(30),30)
140 |BANKWRITE ,@R%,A$,N
150 A$="BALU Thierry 555-44-77":N=3            ' enreg no 3
160 A$=LEFT$(A$+SPACE$(30),30)
170 |BANKWRITE ,@R%,A$,N
180 '----------------------- LECTURE
190 |BANKOPEN ,50
200 '
210 INPUT "QUEL NUMERO ENREG (0 POUR FIN) ";N
220 IF N=0 THEN END
230 B$=SPACE$(30)
240 R%=0
250 |BANKREAD ,@R%,B$,N
260 PRINT B$,R%
270 GOTO 210

QUEL NUMERO ENREG (0 POUR FIN) ? 2
MARTIN Daniel 333-44-56             2
QUEL NUMERO ENREG (0 POUR FIN) ? 1
DUPONT Jean 044-22-63               1
QUEL NUMERO ENREG (0 POUR FIN) ? *Break*
```

BANKFIND ,@code retour,chaîne,début recherche,fin recherche

Donne dans la variable "code retour" le numéro d'enregistrement contenant la chaîne cherchée, si elle existe.

Si la chaîne cherchée n'existe pas, le code retour est égal à −3.

- □ 1 indique un dépassement du bloc de 64 k.
- □ 2 indique une commutation de bloc.

```
175 '  ajouter instructions 10 a 170 du prog precedent
180 '----------------------- RECHERCHE AVEC BANKFIND
190 |BANKOPEN ,50
200 '
210 INPUT "QUEL Nom (enter POUR FIN) ";B$
```

```
220 IF LEN(B$)=0 THEN END
230 R%=0
240 |BANKFIND ,@R%,B$,0,100        ' entre 0 et 100
250 IF R%<0  THEN PRINT "N'EXISTE PAS",R%:GOTO 210
260 N=R%
270 B$=SPACE$(30)
280 |BANKREAD ,@R%,B$,N
290 PRINT B$,N
300 GOTO 210
```

```
QUEL Nom (enter POUR FIN) ? MART
MARTIN Daniel 333-44-56
QUEL Nom (enter POUR FIN) ? DUP
DUPONT Jean 044-22-63
QUEL Nom (enter POUR FIN) ?
```

SAUVEGARDE D'ÉCRAN

La mémoire supplémentaire de 64 k peut être utilisée pour sauvegarder rapidement la mémoire écran de 16 k (1/2 seconde). Ainsi, quatre écrans différents peuvent être sauvegardés et restitués plus rapidement que s'ils avaient été sauvegardés sur disque.

Avant d'utiliser le bloc mémoire de 64 k, pensez à faire RUN "BANKMAN".

| SCREENCOPY ,écran ou bloc dest,écran ou bloc source

L'écran est référencé par le numéro 1. Les quatre blocs mémoire de sauvegarde sont repérés par les numéros 2, 3, 4, 5. Essayez en mode direct :

```
RUN "BANKMAN"

PRINT "ECRAN NUMERO1"

|SCREENCOPY ,2,1         ' sauvegarde ecran dans bloc 2

CLS                      ' effacement ecran

|SCREENCOPY ,1,2         ' restitution bloc 2 dans ecran
```

Ci-dessous, les commandes sont incluses comme instructions dans un programme.

```
10 '-------------------- essai SCREENCOPY
20 ' faire RUN "BANKMAN" avant
30 '
40 MODE 2
50 FOR i=1 TO 20
60  PRINT "ECRAN NUMERO 1"
70 NEXT i
80 |SCREENCOPY ,2,1      ' sauvegarde ecran dans bloc2
90 FOR tp=1 TO 1000:NEXT tp
100 CLS
110 FOR tp=1 TO 1000:NEXT tp
120 |SCREENCOPY ,1,2   ' bloc2 vers ecran
```

|SCREENCOPY ,zone,écran ou bloc dest,écran ou bloc source

Le paramètre "zone", compris entre 0 et 63 peut être ajouté. Dans ce cas, seule la zone de 256 octets spécifiée est sauvegardée.

Ci-dessous, nous utilisons une boucle pour sauvegarder les 64 zones de 256 octets de l'écran :

```
10 FOR Z=0 TO 63
20  |SCREENCOPY Z,3,1
30 NEXT Z
```

Le programme ci-dessous dessine un cercle plein qui est sauvegardé dans le bloc 5. Un second cercle plein est sauvegardé dans le bloc 4. Les deux cercles sont ensuite affichés alternativement.

```
10 ' essai screencopy
20 ' faire run "bankman" avant
30 '
40 MODE 1:INK 0,1:INK 1,20
50 XC=100:YC=300:R=30:CE=1:GOSUB 170   ' 1er cercle
60 |SCREENCOPY ,5,1                    ' sauvegarde dans 5
70 CLS
80 XC=400:YC=100:R=30:CE=1:GOSUB 170   ' 2eme cercle
90 |SCREENCOPY ,4,1                    ' sauvegarde dans 4
100 '------------- affichage alterne
110 FOR TP=1 TO 1000:NEXT TP
120 |SCREENCOPY ,1,5
130 FOR TP=1 TO 1000:NEXT TP
140 |SCREENCOPY ,1,4
150 GOTO 110
160 '------------------- cercle
170 R2=R*R
180 FOR DX=-R TO R
190  DY=SQR(R2-(DX*DX))
200  PLOT XC+DX,YC+DY:DRAW XC+DX,YC-DY,CE
210  NEXT DX
220 RETURN
```

|SCREENSWAP ,zone,écran ou bloc,écran ou bloc

Permet d'échanger l'écran et un bloc de 16 k. Essayez en mode direct :

```
PRINT "ECRAN NUMERO1"

|SCREENCOPY ,2,1

CLS
PRINT "ECRAN NUMERO 2"

|SCREENSWAP ,1,2
|SCREENSWAP ,1,2
```

COMMANDES ACCESSIBLES SOUS CP/M PLUS

- DIR
- DISCKIT3
- ERA
- RENAME
- TYPE
- PIP
- SETKEYS
- SETLST
- SET
- SETDEF
- SHOW
- PALETTE
- LANGUAGE
- DATE
- SUBMIT
- AMSDOS

Pour accéder à CP/M Plus, vous devez placer la disquette CP/M Plus dans le lecteur de disquette. Ensuite, vous frappez "|CPM". Le fichier ".EMS" est alors chargé en mémoire. S'il existe un fichier "PROFILE.SUB", les commandes de ce fichier sont exécutées automatiquement.

Un type est attribué à la plupart des fichiers. Ceci permet de les regrouper par famille. Les types les plus courants sont :

.BAS	BASIC
.BIN	BINAIRE
.BAK	SAUVEGARDE
.COM	COMMANDES

Pour référencer une famille de fichiers, il suffit de spécifier "*" à la place du nom. "?" peut remplacer une lettre indéterminée.

Les commandes DIR,ERA,REN,TYPE sont exécutables sans la disquette CP/M. Les autres commandes doivent être présentes sur disque lorsqu'elles sont appelées. Vous pouvez recopier certaines commandes de la disquette CP/M sur votre disquette de programmes (avec FILECOPY CPM 2.2).

L'exécution d'une commande est annulée en frappant CTRL/C.

DIR

Affiche la liste des fichiers d'une disquette.

`DIR`	affiche la liste de tous les fichiers de la disquette du lecteur A:
`DIR *.BAS`	affiche la liste des fichiers du type ".BAS"
`DIRSYS`	affiche la liste des fichiers systèmes
`DIR [FULL]`	affiche la taille des fichiers

```
    Name       Bytes  Recs   Attributes      Name       Bytes  Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
AMSDOS   COM     1k      8 Dir RW       BANKMAN  BAS     1k      7 Dir RW
BANKMAN  BIN     2k     12 Dir RW       C10CPM3  EMS    25k    200 Dir RW
DATE     COM     3k     23 Dir RW       DEVICE   COM     8k     58 Dir RW
```

DISCKIT3

Cette commande permet de formater, de copier ou de vérifier une disquette (pour une copie d'un seul fichier, cf. FILECOPY CP/M 2.2).

Le formatage ayant déjà été traité dans la partie initiation, utilisons cette commande pour copier une disquette sur une autre (vierge ou non). Placez la disquette CP/M Plus dans le lecteur et frappez "DISCKIT3" :

A>DISCKIT3

Un premier menu apparaît. Placez la disquette source à copier dans le lecteur.

copy	7
format	6
verify	1
exit	0

- Appuyez sur la touche "f7" qui vous permet de choisir le mode "copy".
- Le message suivant apparaît :

Y	copy

- Appuyez sur "Y" (Y=yes) pour confirmer.
- Les quinze premières pistes de la disquette source sont alors lues en mémoire centrale.
- Le message "INSERT DISC TO WRITE" est envoyé. Placez alors la disquette "destination" dans le lecteur. Le système transfère les quinze premières pistes.
- Le programme vous demande de placer à nouveau la disquette source dans le lecteur pour lire les quinze pistes suivantes de la disquette source (INSERT DISC TO READ), etc.
- Lorsque la copie est terminée, le système envoie le message.

COPY COMPLETED
REMOVE DISC
PRESS ANY KEY

Enlevez la disquette et appuyez sur une touche quelconque.

Si vous désirez effectuer une autre copie, appuyez sur "Y", sinon appuyez sur une touche quelconque pour revenir au menu et appuyez sur la touche "f0" pour terminer.

ERA nom fichier

Efface un fichier ou un groupe de fichiers.

`ERA XX.BAS`	efface le fichier XX.BAS
`ERA *.BAK`	efface tous les fichiers du type .BAK
`ERA *.*`	efface tous les fichiers

RENAME nouveau nom=ancien nom

Change le nom d'un fichier.

```
RENAME PROFILE.SUB=PROFILE.ENG
```

Le fichier PROFILE.ENG devient PROFILE.SUB.

TYPE nom fichier

Affiche à l'écran le fichier spécifié.

```
TYPE KEYS.CCP
```

PIP nom fichier destination,nom fichier source

Copie un ou plusieurs fichiers (si vous n'avez qu'un seul lecteur, PIP ne permet pas le transfert d'une disquette sur une autre).

```
PIP XX=KEYS.CCP          copie le fichier KEYS.CCP
                         sous le nom de XX.
```

Les périphériques sont considérés comme des fichiers. Par exemple :

```
PIP LST:=KEYS.CCP
```

édite sur imprimante le contenu du fichier KEYS.CCP.

Les noms des périphériques sont :

LST:	imprimante
CON:	clavier et écran
AUX:	entrée et sortie auxiliaires
PRN:	imprimantes avec options (tabulation, saut de page).

Pour entrer des commandes dans un fichier, vous définissez le clavier comme fichier source avec "CON:". Après avoir frappé la commande PIP, vous frappez les commandes à placer dans le fichier.

Lorsque vous appuyez sur la touche RETURN à la fin de la commande, il n'y a pas de saut de ligne mais seulement un retour en début de ligne. Appuyez sur CONTROL/J pour obtenir un saut de ligne.

CONTROL/Z vous permet de définir la fin du texte.

```
>PIP PROFILE.SUB=CON:
 SETKEYS KEYS.CCP  (puis RETURN)
 CTRL/J            (pour saut de ligne)
 DIR
 CTRL/J
 CTRL/Z            (fin de fichier)
```

PIP peut être utilisé d'une autre façon. Vous frappez PIP puis RETURN. Le système envoie un "*". Vous frappez alors le nom du fichier destination et du fichier source.

```
>PIP
*XX=KEYS.CCP      (puis RETURN)
CONTROL/C
```

SETKEYS nom fichier

Permet de redéfinir certaines touches du clavier. Remplissez avec PIP (ou en BASIC) un fichier KEYS.ESSAI avec :

E128"DIR ↑ M"

et faites SETKEYS KEYS.ESSAI

A chaque fois que vous appuyez sur la touche de fonction "f0" (numéro 128), la commande "DIR" est exécutée.

La disquette CP/M Plus contient un fichier KEYS.CCP.

```
A>TYPE KEYS.CCP
0 N S C "^'#1F'" CCP cursor up
1 N S "^F" cursor right
1 C "^'#9F'"
2 N S C "^'#1E'" cursor down
8 N S "^A" cursor left
8 C "^'#9E'"
9 N S C "^W" copy
16 N S "^G" clr
16 C "^K"
18 C "^E" enter
66 N S "^'27'" esc
66 C "^C"
79 C "^X" del
E #8C "^R" ctrl enter
E #9E "^F^B"
E #9F "^F^B^B"
```

Si vous avez exécuté la commande SETKEYS KEYS.CCP, les touches ont les fonctions suivantes :

CONTROL/A ou <-	curseur à gauche
CONTROL/B ou CONTROL/<-	curseur en début de ligne
CONTROL/C ou CONTROL/ESC	arrêt exécution
CONTROL/E ou CONTRL/RETURN	retour en début de ligne

CONTROL/F ou ->	curseur à droite
CONTROL/G ou CLR	supprime le caractère sous le curseur
CONTROL/H ou DEL	supprime le caractère à gauche du curseur
CONTROL/I ou TAB	curseur à la tabulation suivante
CONTROL/J	valide la ligne frappée
CONTROL/K ou CONTROL/CLR	supprime la fin de la ligne
CONTROL/M ou RETURN	valide la ligne frappée
CONTROL/P	copie sur imprimante et annulation (en bascule)
CONTROL/Q	Reprise de l'affichage (suspendu par CTRL/S)
CONTROL/R ou CONTROL/ENTER	refrappe la commande
CONTROL/S ou ESC	suspend l'affichage
CONTROL/W ou COPY	affiche la dernière commande frappée
CONTROL/X ou CONTROL/DEL	supprime les caractères du début de ligne jusqu'au curseur
CONTROL/Z	fin de texte.

SETLST nom fichier

Permet d'initialiser l'imprimante en lui envoyant des codes. Le fichier spécifié doit contenir :

↑ 'code caractère'
ou ↑ caractère
ou ↑ nom de code (FF, ESC,...).

Par exemple, sur EPSON, pour obtenir des caractères condensés, il faut envoyer à l'imprimante le caractère de code 15 (↑ '15'). Si le fichier contenant le caractère s'appelle "CONDENSE", frappez la commande : SETLST CONDENSE.

SET nom fichier

Change l'attribut des fichiers.

SET *.COM [RO]	Les fichiers .COM ne peuvent être modifiés ou détruits (Read Only)
SET *.COM [RW]	Les fichiers peuvent être modifiés ou détruits (Read Write)
SET *.COM [SYS]	Les fichiers .COM n'apparaissent plus qu'avec DIRSYS.
SET *.COM [DIR]	Les fichiers .COM apparaissent avec la commande DIR.

SETDEF [PAGE] ou [NOPAGE]

Le défilement à l'écran se fait par page ou en continu.

SHOW

Affiche les caractéristiques de la disquette.

```
SHOW [DRIVE]
SHOW [LABEL]
SHOW [USER]
SHOW [DIR]

A>SHOW [DRIVE]

       A: Drive Characteristics
   1,368: 128 Byte Record Capacity
     171: Kilobyte Drive  Capacity
      64: 32 Byte  Directory Entries
      64: Checked  Directory Entries
     128: Records / Directory Entry
       8: Records / Block
      36: Sectors / Track
       2: Reserved  Tracks
     512: Bytes / Physical Record
```

PALETTE fond,écriture

Modifie la couleur d'écriture et de fond (0 à 63).

```
PALETTE 50,1
```

LANGUAGE jeu caractère

Définit le jeu de caractères (AMERICAIN, FRANÇAIS, ANGLAIS,...). LANGUAGE 1 définit le jeu de caractères FRANÇAIS ; le e accent s'obtient avec "{" et "}".

```
LANGUAGE 1     'FRANCAIS
LANGUAGE 3     'ANGLAIS
```

DATE SET
DATE

DATE SET permet de définir le jour et l'heure.
DATE affiche la date.

SUBMIT nom fichier

Exécute les commandes définies dans un fichier.

```
SUBMIT PROFILE.ENG
```

AMSDOS

Provoque le retour à BASIC.

CRÉATION D'UN FICHIER PROFILE.SUB

Lorsque CP/M Plus est active, les commandes du fichier PROFILE.SUB (s'il existe) sont exécutées. Le fichier PROFILE.SUB peut être créé en BASIC ou avec PIP ou bien encore avec un éditeur.

CP/M 2.2

Une disquette CP/M 2.2 est livrée avec le CPC 6128 (cf. chapitre "Le CPC 664").

Cette disquette contient notamment la commande "FILECOPY" qui permet de recopier un fichier d'une disquette sur une autre avec un seul lecteur.

La commande "DISCKIT2" est l'équivalente de "DISCKIT3". Elle permet de formater et de copier une disquette sur une autre.

La commande "CLOAD" permet de recopier un fichier de données (ASCII) provenant d'une cassette. On pourra également utiliser le programme ci-dessous pour effectuer le transfert.

```
10 '---------- TRANSFERT CASSETTE ---> DISQUE
20 INPUT "Nom fichier ";NF$
30 |TAPE.IN
40 OPENIN ""
50 OPENOUT NF$
60 '
70 IF EOF=-1 THEN CLOSEOUT:CLOSEIN:END
80 LINE INPUT #9,lig$
90 PRINT #9,lig$
100 GOTO 70
```

Sur CPC 664, la disquette CP/M 2.2 est fournie avec FORMAT et DISCCOPY.

INSTRUCTIONS BASIC AMSTRAD CPC 6128 et 664

- PEN
- PLOT
- PLOTR
- DRAW
- DRAWR
- MOVE
- MOVER
- FILL
- FRAME
- MASK
- GRAPHICS PAPER
- GRAPHICS PEN
- CUSOR
- COPYCHR$
- DERR
- ON BREAK CONT
- DEC$

Certaines instructions du BASIC CPC 464 ont été améliorées : PEN, PLOT, PLOTR, DRAW, DRAWR, MOVE, MOVER.

Des instructions ont été ajoutées : FILL, FRAME, MASK, GRAPHICS PEN, GRAPHICS PAPER, CURSOR, COPYCHR$, DERR, ON BREAK CONT, DEC$.

PEN #fenêtre,stylo,mode

Le paramètre "mode" a été ajouté à l'instruction PEN. Si "mode" est égal à 1, la fonction PAPER est inhibée.

Ci-dessous, le message "AMSTRAD" est affiché sur la couleur de fond spécifiée par PAPER si "T" est égal à 0. En revanche, si "T" est égal à 1, la couleur de fond ne change pas.

```
10 '---------- MODE TRANSPARENT
20 MODE 1
30 INK 0,1:INK 1,24:INK 2,20:INK 3,6
40 INPUT "0 OU 1 ";T
50 PAPER 2
60 PEN 1,T
70 PRINT "AMSTRAD 664"
80 PAPER 3
90 PEN 1,T
100 PRINT "AMSTRAD 664"
110 PAPER 0
```

PLOT X,Y,stylo,opération
PLOTR DX,DY,stylo,opération
DRAW X,Y,stylo,opération
DRAWR DX,DY,stylo,opération

Le paramètre "opération" a été ajouté. Une opération XOR,AND ou OR est effectuée avant le tracé.

0 : nul
1 : XOR (OU exclusif)
2 : AND (ET)
3 : OR (OU)

Par exemple, si XOR (OU exclusif) est spécifié, l'ancien tracé (s'il existait) disparaît. La ligne 40 du programme ci-dessous trace une droite. La ligne 60 trace la même droite avec une opération XOR. Par conséquent, le tracé disparaît.

```
10 '----- OR,XOR,AND
20 MODE 1
30 INK 0,1:INK 1,24
40 PLOT 100,100,1:DRAW 200,200,1
50 FOR TP=1 TO 1000:NEXT TP
60 PLOT 100,100:DRAW 200,200,1,1
70 GOTO 50
```

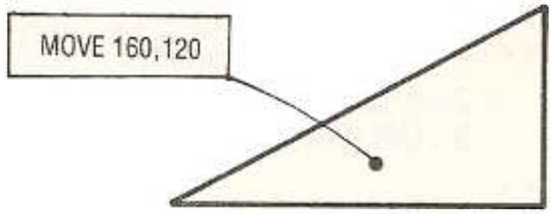
MOVE X,Y,stylo,opération
MOVER DX,DY,stylo,opération

Les paramètres "stylo" et "opération" ont été ajoutés aux instructions "MOVE" et "MOVER".

FILL stylo

Remplit une figure avec la couleur du stylo spécifiée. Le curseur doit être positionné à l'intérieur de la figure avec MOVE ou MOVER. Le "stylo" spécifié dans MOVE doit être celui qui a servi au tracé de la figure. En revanche, le stylo spécifié dans FILL peut être différent.

```
10 '--------- FILL
20 MODE 1
30 CE=1    ' Couleur ecriture
40 INK 0,26:INK 1,0:INK 2,20
50 PLOT 100,100,CE
60 DRAW 200,100,CE
70 DRAW 200,200,CE
80 DRAW 100,100,CE
90 '----- REMPLISSAGE TRIANGLE
100 '
110 MOVE 160,120,CE      ' Positionnement
120 C=1                  ' Stylo remplissage
130 FILL C
140 '
150 ' La couleur de remplissage peut etre differente de la couleur
    du contour
160 ' ex:C=2
170 '
180 'Essayer en ajoutant:125 ink c,1,24
```



Le programme ci-dessous dessine un cercle et remplit la moitié inférieure.

```
10 '-------- CERCLE
20 MODE 1
30 INK 0,26:INK 1,0:INK 2,20
40 XC=200:YC=200        ' CENTRE
50 R=60                 ' RAYON
60 '----- CERCLE
70 PLOT XC+R,YC
80 FOR A=0 TO 2*PI+0.2 STEP 0.2
90  X=XC+R*COS(A)
100  Y=YC+R*SIN(A)
110  DRAW X,Y,1
120 NEXT A
130 '------- DIAMETRE
140 PLOT XC-R,YC:DRAWR R*2,0,1
150 '-------- REMPLISSAGE DEMI CERCLE
160 MOVER -5,-2:FILL 2
```

Ci-dessous, nous représentons un demi-cercle plein en traçant un cercle "invisible" ; la couleur d'écriture du stylo 2 est la même que celle du papier.

Ensuite, nous remplissons la moitié supérieure du cercle.

```
10 '--------- DEMI CERCLE PLEIN
20 MODE 1
30 INK 0,1:INK 1,24:INK 2,1
40 XC=200:YC=200
50 R=60
60 '------- CERCLE INVISIBLE
70 PLOT XC+R,YC
80 FOR a=0 TO 2*PI+0.2 STEP 0.2
90  X=XC+R*COS(A)
100  Y=YC+R*SIN(A)
110  DRAW X,Y,2
120 NEXT a
130 '-------- DIAMETRE
140 PLOT XC-R,YC:DRAWR R*2,0
150 '-------- REMPLISSAGE
160 MOVER -5,+2:FILL 1
```

Le programme "dessinateur" de la page 129 peut être amélioré en lui ajoutant l'instruction 355.

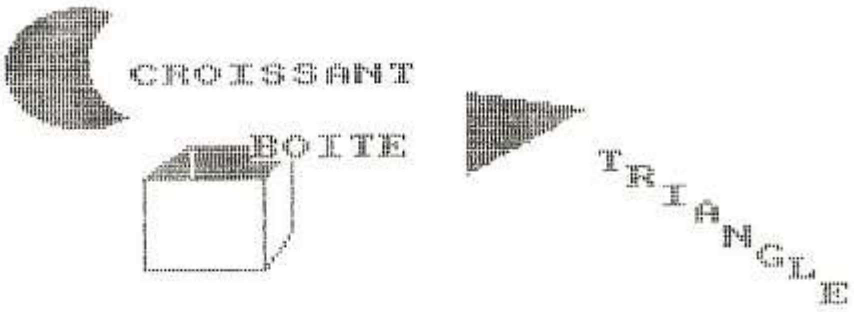
En positionnant le curseur à l'intérieur d'une figure et en appuyant sur "P", vous la remplissez.

```
1 '------- AJOUT PROGRAMME DESSINATEUR P 129 (CPC664)
355 IF C$="P" THEN MOVE X,Y,CE:FILL CE
356 IF C$="S" THEN SAVE "DES",B,&C000,&4000
357 IF C$="L" THEN LOAD "DES"
```

Attention ! Le curseur doit avoir la couleur de la figure.

Pour mélanger texte et graphique, ajouter les lignes 270 et 275. En frappant "#", vous passez alternativement en mode texte et en mode graphique.

```
274 IF C$="#" THEN TT=ABS(TT-1):GOTO 150
275 IF TT=1 THEN TAG:PLOT X,Y,1:PRINT C$;:X=X+16:TAGOFF:GOTO 150
```



FRAME

Lorsqu'un caractère est affiché avec TAG, les "points" du caractère ne sont pas affichés simultanément. FRAME permet de synchroniser l'affichage des points.

```
10 '---------- FRAME
20 MODE 1
30 INK 0,1:INK 1,24
40 GRAPHICS PAPER 0
50 TAG
60 FOR X=0 TO 500 STEP 4
70  FRAME      ' SYNCHRONISE
80  MOVE X,200,1
90  PRINT CHR$(32);CHR$(143);
100 NEXT X
```

Essayez le programme ci-dessus sans FRAME.

MASK masque,premier point

Permet de tracer des pointillés suivant le "masque" spécifié. Si "masque" est égal à "11110000", 4 points sur 8 seulement sont affichés.

```
10 '---------- MASK
20 MODE 1
30 INK 0,26:INK 1,0
40 GRAPHICS PAPER 0        Initialisation
50 '
60 MASK &X11110000,1
70 MOVE 100,100,1:DRAW 200,100,1
80 MASK &X11110000,0
90 MOVE 100,120,1:DRAW 200,120,1
100 '
110 '  L'instruction 40 est necessaire
120 ' si GPAPER a deja ete utilise
```

Si "premier point" est égal à 1, il est affiché. Si "premier point" est égal à 0, il n'est pas affiché.

GRAPHICS PAPER papier

Détermine la couleur du papier pour les graphiques ou le texte affichés avec TAG. Ci-dessous, les pointillés sont affichés sur fond bleu.

```
10 '-------- GRAPHICS PAPER
20 MODE 1
30 PAPER 0:PEN 1
40 INK 0,26:INK 1,0:INK 2,20
50 '
60 MASK &X11100000
70 GRAPHICS PAPER 2
80 MOVE 100,100,1:DRAW 200,200,1
90 '--
100 TAG:PRINT "COUCOU";
```

GRAPHICS PEN stylo,mode

Détermine le stylo par défaut pour les instructions graphiques. Si "mode" est égal à 1, l'effet de GRAPHICS PAPER est annulé.

```
10 '---------- GRAPHICS PEN
20 MODE 1
30 PAPER 0:PEN 1
40 INK 0,1:INK 1,24:INK 2,20
50 '
60 GRAPHICS PEN 2
70 PLOT 100,100:DRAW 200,200
80 '
90 LOCATE 10,10:PRINT "TEXTE"
```

Le programme ci-dessous affiche "TEXTE" sur la couleur du papier 2. Essayez le programme avec 50 GRAPHICS PEN 1,1.

```
10 '------------ GRAPHICS PAPER ET PEN
20 MODE 1
30 INK 0,1:INK 1,0:INK 2,20
40 GRAPHICS PAPER 2
50 GRAPHICS PEN 1
60 MOVE 100,100
70 TAG
80 PRINT "TEXTE";
90 '
100 ' ESSAYER AVEC:50 GRAPHICS PEN 1,1
```

CURSOR mode

Lorsque vous utilisez INKEY$, le curseur n'apparaît que si vous avez programmé "CURSOR 1". "CURSOR 0" fait disparaître le curseur.

```
10 '------------ CURSOR
20 CLS
30 LOCATE 10,10           CURSEUR EN 10,10
40 CURSOR 1               CURSEUR APPARENT
50 C$=INKEY$:IF C$="" THEN 50
60 IF ASC(C$)=13 THEN END
70 PRINT C$;
80 GOTO 40
```

COPYCHR$ (#fenêtre)

Fournit le caractère à l'écran sous le curseur.

```
10 '--------- COPYCHR$
20 CLS
30 LOCATE 10,10:PRINT "ABCDE"
40 '
50 LOCATE 10,10:X$=COPYCHR$(#0)
60 '
70 LOCATE 20,20:PRINT X$
```

Le programme ci-dessous recopie sur imprimante le texte affiché à l'écran (en MODE 2).

```
10 '-------- COPIE ECRAN(TEXTE)
20 FOR L=1 TO 24
30  FOR C=1 TO 80
40   LOCATE C,L
50   C$=COPYCHR$(#0)
60   PRINT #8,C$;
70  NEXT C
80  PRINT #8
90 NEXT L
```

DERR

Fournit les codes erreur pour le disque.

```
10 ON ERROR GOTO 50
20 OPENIN "XX"
30 END
40 '-------- Analyse erreur
50 PRINT "ERR=";ERR;"erl=";ERL
60 PRINT "DERR=";DERR
70 IF DERR=146 THEN PRINT "FICHIER NON TROUVE":END
```

ON BREAK CONT

Empêche l'interruption de l'exécution d'un programme si l'opérateur appuie sur la touche ESC. Est annulé par ON BREAK STOP.

```
10 ON BREAK CONT
20 '
30 FOR I=1 TO 100
40   PRINT "APPUYER SUR ESC"
50 NEXT I
```

DEC$ (nombre,format)

Fournit une chaîne de caractères avec le format spécifié.

```
10 '-------- DEC$
20 X=5
30 X=67.8
40 X$= DEC$(X,"####.##")
50 PRINT X$
60 '
70 'CF PRINT USING

   67.80
```

ADAPTATION DES PROGRAMMES P143,148,153

Le mode "FIN" ne provoque pas l'arrêt du programme après la sauvegarde des tables afin de permettre d'effectuer plusieurs sauvegardes sur cassette.

Pour une sauvegarde sur disquette, vous pouvez faire :

```
P143: 160 IF M$="FIN" THEN GOSUB 500:END
P148: 190 IF M$="FIN" THEN GOSUB 790:END
P153: 200 IF M$="FIN" THEN GOSUB 980:END
```

Sur CPC 664, LINE INPUT envoie un point d'interrogation après le message (contrairement au CPC 464).

LISTE DES ERREURS DISQUE

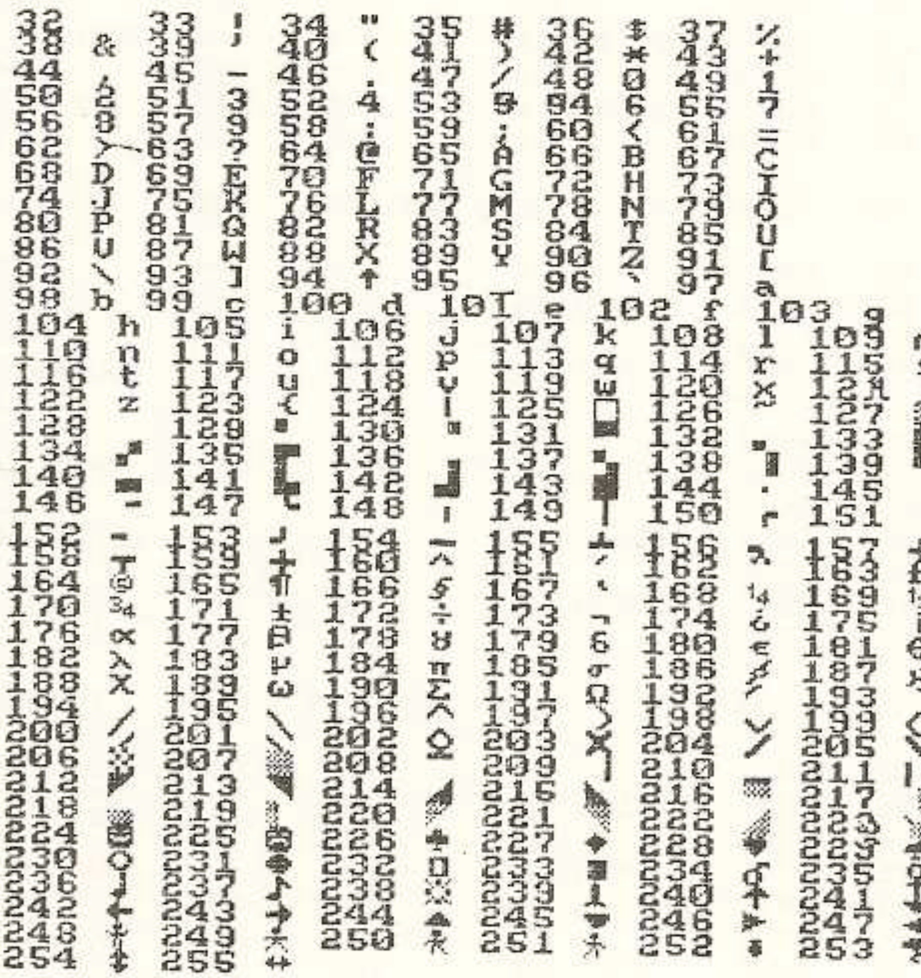
142 : état du canal non valable
143 : fin de fichier physique
144 : nom de fichier incorrect
145 : le fichier existe déjà
146 : le fichier n'existe pas
147 : le catalogue est plein
148 : la disquette est pleine
149 : la disquette a été changée avec des fichiers ouverts
150 : le fichier peut seulement être lu
154 : fin de fichier logique

Lorsqu'une erreur disque se produit, la variable ERR prend la valeur 32. Pour la fin de fichier, nous avons obtenu sur 6128 ERR=24 et DERR=0.

programme

ANNEXE | 1

CODES ASCII



programme

ANNEXE | 2

TABLE DES NOTES

Octave:-1	FREQ	PERIODE	Octave: 2	FREQ	PERIODE
DO	131	956	DO	1047	119
DO DIESE	129	967	DO DIESE	1034	121
RE	147	851	RE	1175	106
MI BEMOL	156	804	MI BEMOL	1245	100
MI	165	758	MI	1319	95
FA	175	716	FA	1397	89
FA DIESE	185	676	FA DIESE	1480	84
SOL	196	638	SOL	1568	80
SOL DIESE	208	602	SOL DIESE	1661	75
LA	220	568	LA	1760	71
SI BEMOL	233	536	SI BEMOL	1865	67
SI	247	506	SI	1976	63

Octave: 0	FREQ	PERIODE	Octave: 3	FREQ	PERIODE
DO	262	478	DO	2093	60
DO DIESE	258	484	DO DIESE	2067	60
RE	294	426	RE	2349	53
MI BEMOL	311	402	MI BEMOL	2489	50
MI	330	379	MI	2637	47
FA	349	358	FA	2794	45
FA DIESE	370	338	FA DIESE	2960	42
SOL	392	319	SOL	3136	40
SOL DIESE	415	301	SOL DIESE	3322	38
LA	440	284	LA	3520	36
SI BEMOL	466	268	SI BEMOL	3729	34
SI	494	253	SI	3951	32

Octave: 1	FREQ	PERIODE	Octave: 4	FREQ	PERIODE
DO	523	239	DO	4186	30
DO DIESE	517	242	DO DIESE	4135	30
RE	587	213	RE	4699	27
MI BEMOL	622	201	MI BEMOL	4978	25
MI	659	190	MI	5274	24
FA	698	179	FA	5588	22
FA DIESE	740	169	FA DIESE	5920	21
SOL	784	159	SOL	6272	20
SOL DIESE	831	150	SOL DIESE	6645	19
LA	880	142	LA	7040	18
SI BEMOL	932	134	SI BEMOL	7459	17
SI	988	127	SI	7902	16

programme

ANNEXE | 3

MESSAGES D'ERREURS

Si le message d'erreur ne suffit pas pour détecter d'où provient l'erreur, on pensera à **visualiser les valeurs des variables en "mode direct"**. Ceci aidera bien souvent à la retrouver.

Des instructions STOP judicieusement placées permettront également de mieux suivre l'évolution des valeurs des variables. On pourra ainsi insérer momentanément des instructions de visualisation des valeurs des variables (PRINT "X=";X par exemple).

La **trace** (avec **TRON**), un peu trop riche en informations, ne sera utilisée que dans les cas particulièrement délicats.

Pour obtenir le code erreur, on fera **"PRINT ERR"** en mode direct.

Message	Code	
ARRAY ALREADY DIMENSIONNED	10	(tableau redimensionné) Un tableau est à nouveau dimensionné ou une dimension de tableau est déclarée pour un tableau non déclaré explicitement mais créé par BASIC (avec une dimension 10) parce qu'il a déjà été référencé (par une instruction A(4)=X par exemple).
CANNOT CONTINUE	17	(l'exécution ne peut se poursuivre) On a tenté de poursuivre l'exécution d'un programme qui : – a été stoppé à cause d'une erreur – a été modifié après une interruption.

Message	Code	
DATA EXHAUSTED	4	(DATAs épuisés) Un READ est exécuté alors qu'il n'y a plus de DATA à lire. – On a oublié des DATAs – On a oublié de programmer RESTORE.
DIVISION BY ZERO	11	(division par zéro)
DIRECT COMMAND FOUND	21	(commande directe) Une commande directe sans numéro de ligne est trouvée lors d'un chargement.
EOF MET	24	(fin de fichier rencontré) Prévoir un test de fin de fichier avec "EOF" avant INPUT #9.
FILE ALREADY OPEN	27	(fichier déjà ouvert)
FILE TYPE ERROR	25	(mauvais type de fichier) On tente d'ouvrir un fichier programme avec OPENIN.
INVALID DIRECT COMMAND	12	(illégal en mode direct) L'instruction frappée n'est pas validée en mode direct ; elle ne peut être exécutée que précédée d'un numéro de ligne.
IMPROPER ARGUMENT	5	(argument invalide) Un paramètre hors du domaine normal à une fonction arithmétique ou à une fonction chaîne. EX. : – argument négatif pour SQR – longueur spécifiée dans LEFT$, MID$, RIGHT$ non compris entre 0 et 255.
LINE DOES NOT EXIST	8	(numéro de ligne indéfini) Une instruction référence une ligne qui n'existe pas.
LINE TOO LONG	23	(ligne trop longue) Une ligne de programme dépasse 255 caractères.
NEXT MISSING	26	(FOR sans NEXT) Une instruction n'a pas de NEXT associé.

Message	Code	
MEMORY FULL	7	(plus de mémoire centrale) Il n'y a plus assez de place en mémoire centrale. – On doit supprimer une partie du programme ou les tableaux. – La structure des boucles est trop compliquée.
OPERAND MISSING	22	(opérande manquant) Il manque un opérande dans une expression.
OVERFLOW	6	(dépassement de capacité) On a tenté de donner à une variable entière une valeur non comprise entre −32768 et +32767.
RESUME MISSING	19	(pas d'instruction RESUME) Une instruction END a été rencontrée dans un programme de traitement d'erreur.
SUBSCRIPT OUT OF RANGE	9	(référence en dehors du domaine) Un tableau est référencé en dehors de ses dimensions. Souvent, pour un tableau non déclaré qui a été dimensionné à 10 par BASIC parce que référencé (par une instruction A(4)=X par exemple).
SYNTAX ERROR	2	(erreur de syntaxe) La ligne contient une erreur de syntaxe : – parenthèses non appairées – ponctuation incorrecte – instruction n'existant pas – etc.
TYPE MISMATCH	13	(désaccord entre numérique et chaîne) Une valeur numérique est affectée à une chaîne ou l'inverse.
UNEXPECTED NEXT	1	(NEXT sans FOR) Une variable dans un NEXT ne correspond à aucun FOR. Ex. : La ligne FOR correspondante a été effacée sans le NEXT associé.
UNEXPECTED RESUME	20	(RESUME sans ON ERROR)

Message	Code	
UNEXPECTED RETURN	3	(RETURN sans GOSUB) – On est "entré" dans un sous-programme par GOTO (au lieu de GOSUB). – On est "entré" dans un sous-programme par erreur parce que l'on a oublié STOP ou END à la fin de l'exécution de son programme (devant un sous-programme).
UNKNOW COMMAND	28	(commande inconnue)
UNKNOW USER FONCTION	18	(fonction utilisateur indéfinie) Une fonction FN est appelée avant d'être définie par DEF FN.
WEND MISSING	29	(WEND manquant)

ANNEXE | 4

INDEX

Pour approfondir vos connaissances en BASIC, et mieux connaître le système des CPC 464, 664 et 6128, ainsi que du PCW 8256, P.S.I. vous propose une palette d'ouvrages utiles.

POUR MAITRISER LE BASIC AMSTRAD

- □ **Basic Amstrad, 2. Programmes,** par Jacques Boisgontier (Editions du P.S.I.)
  Pour pratiquer le Basic Amstrad, cet ouvrage donne de nombreux programmes de gestion, d'éducation et de jeu où le rôle des fichiers est expliqué et largement commenté.

- □ **Basic Plus, 80 routines sur Amstrad,** par Michel Martin (Editions du P.S.I.)
  Pour pousser votre Amstrad au maximum de ses capacités : 80 routines de simulation d'instructions qui n'existent pas en Basic Amstrad.

- □ **Périphériques et gestion de fichiers sur Amstrad, CPC 464, 664 et 6128,** par Daniel-Jean David (Editions du P.S.I.)
  Cet ouvrage s'adresse aux lecteurs, déjà initiés à l'Amstrad, souhaitant programmer tous les périphériques des CPC 464, 664 et 6128, et programmer des applications utilisant des fichiers sur cassette ou disquette.

POUR MIEUX CONNAÎTRE LE SYSTÈME DES CPC ET DU PCW 8256

- □ **Clefs pour Amstrad, 1. Système de base,** par Daniel Martin (Editions du P.S.I.)
  Mémento présentant synthétiquement le jeu d'instructions du Z80, les points d'entrée des routines système, les connecteurs et brochages, etc. Le livre de chevet du programmeur sur Amstrad.

- □ **Clefs pour Amstrad, 2. Système disque,** par Daniel Martin et Philippe Jadoul (Editions du P.S.I.)
  Ce deuxième tome consacré au système disque présente les points d'entrée des routines disque, les blocs de contrôle, la programmation et les brochages des circuits spécialisés... La deuxième partie du livre est aussi destinée aux possesseurs d'Amstrad 8256.

- □ **CP/M Plus sur Amstrad,** par Yvon Dargery (Editions du P.S.I.)
  Toutes les commandes CP/M et CP/M Plus pour maîtriser le système des 6128 et 8256 : un ouvrage de référence illustré par de nombreux programmes.

- □ **Le livre de l'Amstrad,** tome 1 — par Daniel Martin et Philippe Jadoul (BCM — diffusé par P.S.I.)
  Ce livre, destiné aux programmeurs des CPC 464 et 664, donne une étude complète de tous les circuits internes, et analyse la structure interne du BASIC. Vous y trouverez, en outre, une étude complète des RSX, et des programmes de scrolling, de traçage de rectangles, de coloriage de surface et de manipulation vectorielle.

Achevé d'imprimer en octobre 1988
sur les presses de l'imprimerie Laballery
58500 Clamecy
Dépôt légal : octobre 1988

N° d'impression : 810004
N° d'édition : 86595-230-6
ISBN : 2-86595-230-4

BASIC AMSTRAD CPC 464, 664 ET 6128

1. MÉTHODES PRATIQUES

Cet ouvrage s'adresse à vous qui, de plus en plus nombreux, avez déjà pratiqué le BASIC et qui souhaitez pousser votre Amstrad à son meilleur niveau de performance.

Vous découvrirez, grâce à de nombreux programmes-exemples, toutes les instructions des CPC 464, 664 et 6128 au fur et à mesure de vos besoins, ainsi que les avantages spécifiques à votre ordinateur : jeu d'instructions très complet, gestion des interruptions en BASIC, sortie stéréo ou haut-parleur intégré, etc.

La deuxième partie comprenant des programmes de graphisme et de gestion fera ressortir la richesse de la palette de couleurs en haute résolution et la rapidité d'exécution de l'Amstrad.



9 782865 952304

PRIX : 105 FF

ÉDITIONS P.S.I.
DIFFUSÉ PAR P.C.V. DIFFUSION
9, RUE MÉCHAIN - 75680 PARIS CEDEX 14

ISBN : 2-86595-230-4
500072

PHOTO : L. MAUNOURY